PCI Express® Architecture Link Layer and Transaction Layer

Test Specification

Revision 3.0

June 6, 2013

REVISION HISTORY

Rev	Version	History	Date
2.0	1.0	Initial Release	08/11/2008
3.0	1.0	Updated to reflect PCI Express 3.0 , Rev 3,0 Release	06/06/2013

TABLE OF CONTENTS

LIST OF FIGURES

1

1. Introduction

This test specification primarily covers testing of PCI Express Device and Port types for compliance with the link layer and transaction layer requirements of the PCI Express Base Specification. Device and Port types that do not have a link (e.g., Root Complex Integrated Endpoints, Root Complex Event Collectors) are not tested under this test specification. At this point, this test specification does not describe the full set of PCI Express tests for all link layer or transaction layer requirements.

One part of this test specification provides a list of definitions pertaining to the link layer. Another part of this document contains a list of test definitions pertaining to the transaction layer. Test descriptions provide more detailed information (algorithm, test set up, results interpretation, etc.) on how devices are tested.

In addition to the test definitions specified in this test specification, devices must also meet other applicable requirements described in the latest versions of the following documents as well as any other criteria required by the PCI-SIG:

PCI Express Architecture PHY Test Specification

PCI Express Architecture Configuration Space Test Specification

Platform BIOS Test Considerations for PCI Express Architecture

1.1 Abbreviations

BAR – Base Address register

DUT – Device Under Test

KEP – Known End Point

PTC – Protocol Test Card

2

2. General Testing Overview and Topologies

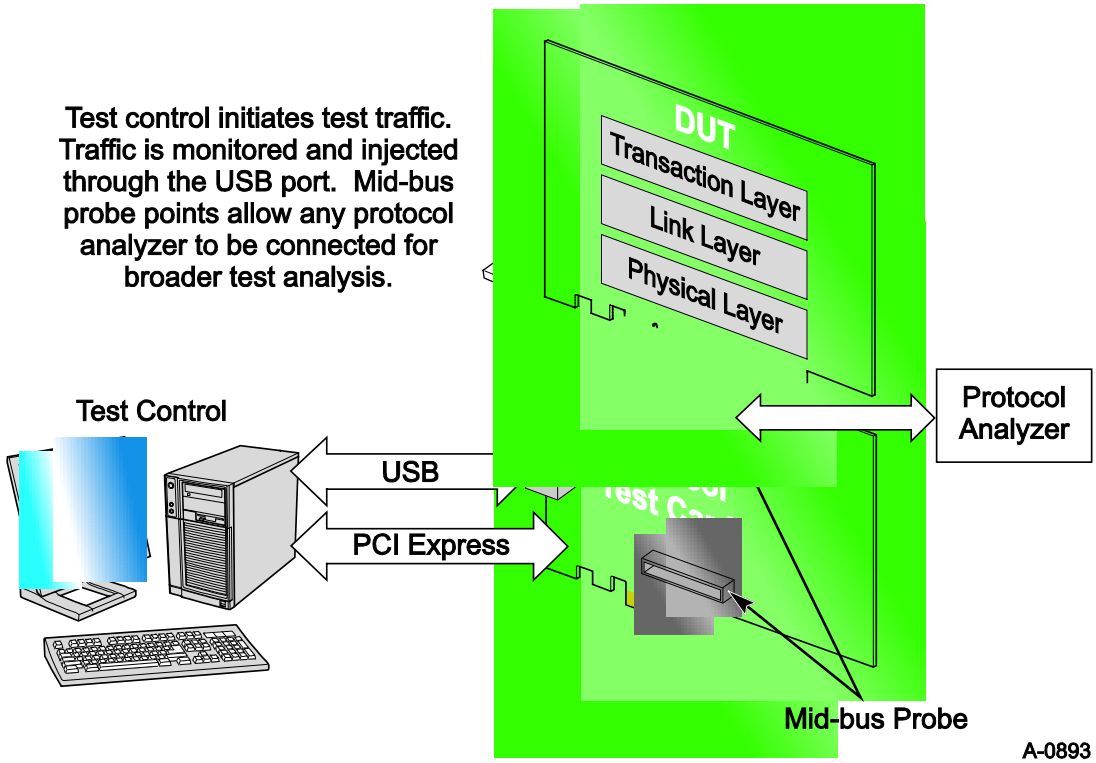

Figure 2–1. Transaction Layer Basic Test Topology

Figure 2–1 shows the connection between the PTC card and the test control system.

Advanced Error reporting is an optional capability documented in the PCI Express Base Specification. However, if a DUT implements such capability, it must meet the requirements laid out in the PCI Express Base Specification. Hence in the test definitions, the check for advanced error reporting requirements shall be applicable only if the DUT implements it.

In order to test at the link layer or the transaction layer, it is assumed that the test equipment will have the following capabilities:

1. The test equipment allows any DUT with up to x16 PCI Express lane width to be connected to the golden system, supporting either non-reversed or reversed lane ordering. (The test equipment is not required to physically route more than x1 PCI Express lanes, but it is not precluded from doing so.)

2. If the test equipment physically routes more than x1 PCI Express lanes, it must support all link widths between x1 and xN (where N is the number of physically routed lanes), allowed under the specific PCI Express Base specification revision.

3. The test equipment supports all data rates allowed under the appropriate revision of the *PCI Express Base Specification* (see Section 3, *Test Descriptions*).

4. The test equipment supports any necessary Link Equalization procedures, defined in the appropriate revision of *PCI Express Base Specification*.

5. The DUT will operate normally (though perhaps at reduced performance levels) with its drivers loaded and executing all applicable applications with the right configuration.

6. The test equipment has the following minimum capabilities:

    a. Intentionally generate a NAK DLLP for matching TLPs from the DUT.

    b. Intentionally ignore (no ACK or NAK DLLP generated) for matching TLPs from the DUT.

    c. Delay ACK or NAK DLLP generation within both legal and illegal boundaries.

    d. Drop and delay matching TLPs to and from the DUT.

    e. Corrupt different CRCs and TLP Digest fields of TLPs and DLLPs to and from the DUT.

    f. Generate incorrect sequence numbers in ACK and NAK DLLPs to and from the DUT.

    g. Generate and receive traffic from the platform: have basic Master/Target memory and message capabilities.

    h. Generate Set_Slot_Power_Limit messages (under test software control), when the test equipment is capable of supplying more than the minimum defined power from the slot connector.

    i. Give test software control on when the test conditions are to be applied through the ARM and DISARM commands. The test equipment will only apply the test conditions between ARM and DISARM.

    j. Test equipment has Configuration Space that can be exposed to the DUT. The Configuration Space may be populated by test software.

    k. Test equipment must report ARI Forwarding Supported=1 (or be configurable by test software to report this) when configured as a Downstream Port.

l. Test equipment has some memory resources (at least 4 KB) behind one of its BARs (Base Address register) and the test software can use it as either scratch pad memory or trace buffer memory.

m. Get the status from the test equipment on where it is in the process of applying the test conditions. In this document, it is assumed that the test equipment can assert the test conditions multiple times (programmable) and that the status indicates how many times the test equipment still needs to assert them. A count of zero indicates the test equipment has asserted the test conditions.

n. The test equipment can apply similar test conditions to a platform under test or an add-in card under test. Hence it shall have suitable operating modes.

7. The test equipment will support a PCI Express CEM specification compliant slot connector that will accept an edge connector that is x16 width.

8. The test equipment will provide a PCI Express CEM specification compliant edge connector that will fit in a PCI Express slot connector. (The width of the edge connector is not defined, but must be at least x1.)

9. The test equipment will not consume more than the PCI Express CEM specification defined amount of power from the edge connector. Any additional power required by the test equipment or for the slot connector, must be supplied by an external power connector.

10. The test equipment will provide the PCI Express CEM specification defined amount of power to the slot connector.

11. The test equipment will not exceed the physical add-in card volume of a single-size (i.e., one slot wide) add-in card, defined by the PCI Express CEM specification. (Exception: The height dimension of the card may exceed the specification allowed card height in the area where the slot connector is. If, optionally, the slot connector is detachable, then the card minus the detached slot connector must not exceed the specification allowed card height in any area.)

12. The test equipment will ensure that all physical add-in card volumes supported by the PCI Express CEM specification will be able to plug into the slot connector. (This includes supporting triple-size add-in cards that have a single edge connector, but the total volume of three cards.)

While not mandatory, the test equipment may support the following capabilities:

1. The test equipment may tolerate ASPM protocols (e.g., L0s, L1) from the DUT.

Going forward, test equipment with all the above characteristics and capabilities will be referred to as the Protocol Test (PTC) card in this document. In order to facilitate testing of both platforms and add-in cards, the PTC will have the following modes of operation:

1. Platform Test Mode – PTC will apply the test conditions to a Downstream Port DUT (e.g., Root Port, Switch Downstream Port, or PCI/PCI-X to PCI Express Bridge) integrated on a motherboard that it is plugged into. The overall topology (platform + PTC) is referred to as the Platform Test Topology and is shown below in Figure 2–2. In the test descriptions, this configuration is listed under the title: Root Port Test.

2. Add-Card Test Mode – PTC will apply the test conditions to an Upstream Port DUT (e.g., Endpoint, Switch Upstream Port, or PCI Express to PCI/PCI-X Bridge) on an add-in card plugged into the PTC. The topology (platform +PTC + add-in card) is referred to as the Endpoint Test Topology and is shown in Figure 2–3. In the test descriptions, this configuration is listed under the titles: Endpoint Device Test, Switch Upstream Port Test, PCI Express to PCI/PCI-X Bridge Test.

3. Platform Test Mode (add-in Switch card) – PTC will apply the test conditions to a Downstream Port DUT (e.g., Switch Downstream Port, PCI/PCI-X to PCI Express Bridge) on an add-in card plugged into the platform on one end and the PTC on the other. The topology (platform + Add-in card + PTC) is referred to as the Switch Test Topology and is shown in Figure 2–4. In the test descriptions, this configuration is listed under the title: Switch Downstream Port Test, PCI/PCI-X to PCI Express Bridge Test.

The relevant macros for use with PTC are defined in Appendix A. The purpose of the macros is to make the test definition writing consistent and focus on the functionality rather than on any test equipment implementation. Examples of test topology are shown in the figures that follow.

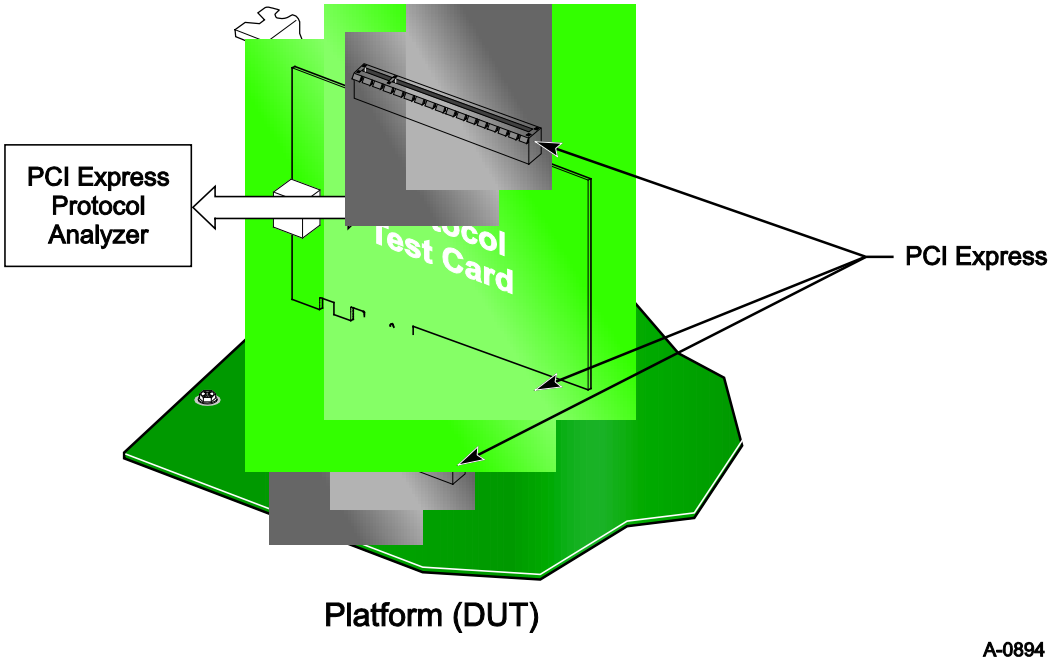

Figure 2–2. Platform Test Topology

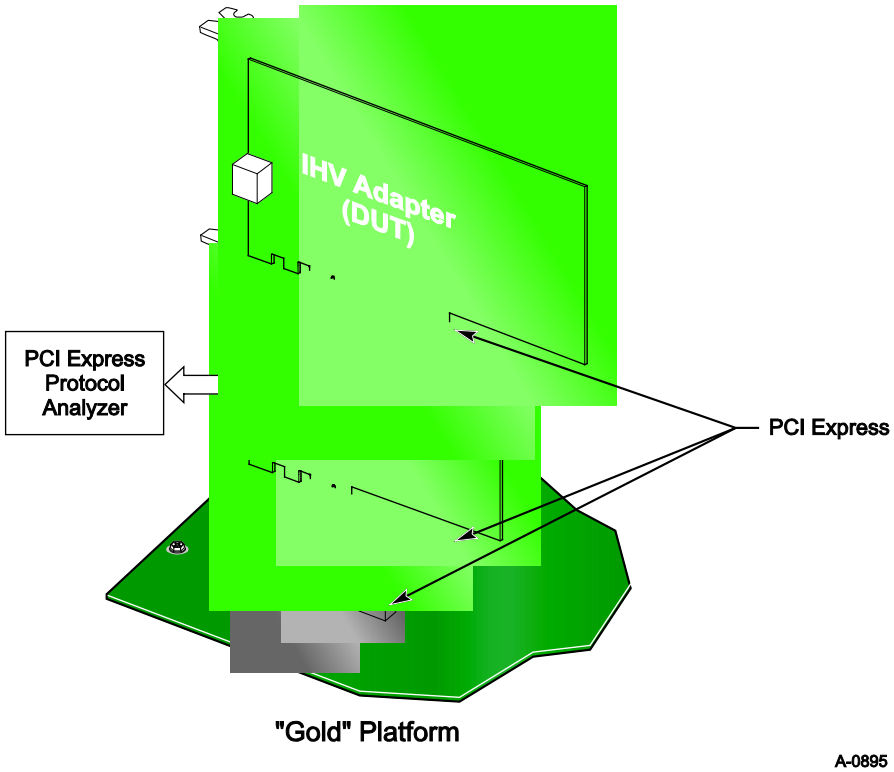

Figure 2–3. Endpoint Test Topology

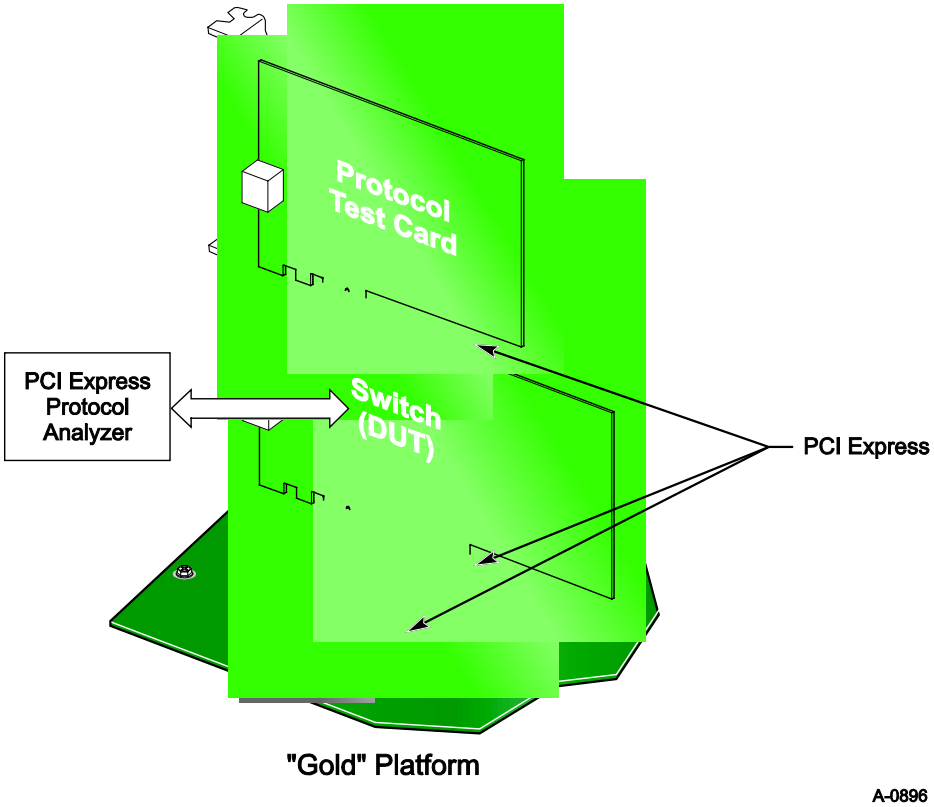

Figure 2–4. Switch Test Topology

3

3. Test Descriptions

All tests are run at 2.5 GT/s for DUTs that support up to 2.5 GT/s, at both 2.5 GT/s and 5.0 GT/s for DUTs that support up to 5.0 GT/s, and at all of 2.5 GT/s, 5.0 GT/s, and 8.0 GT/s for DUTs that support up to 8.0 GT/s.

When testing multi-function DUTs, error reporting shall only be enabled (when required) in the Device Control register and the SERR# Enable bit (Command register) for the DUT function currently being tested, and disabled for all the other functions in the DUT. The error log registers will only be checked in the DUT function currently being tested (if it exists). The platform shall be configured so that it can tolerate receiving any error message from the DUT. Specifically, the Root Port that originates the link hierarchy connected to the DUT shall have its Root Control register programmed so that it will not generate any system errors whenever it receives an error message from the DUT.
When testing multi-function DUTs, any link specific settings will be modified through function 0, regardless of which DUT function is currently being tested.

3.1 Data Link Layer Packet Rules

3.1.1 Test 41-20 ReservedFieldsDLLPReceive

Test Introduction

The intent of this test is to verify that the DUT ignores reserved fields (when they are set to arbitrary data) in an ACK DLLP.

Notes:

1. Test applies to all PCI Express port types.
2. Test could optionally be extended to cover all DLLP types as deemed necessary.
3. DATA_BUF – holds the data read back from the device.

3.1.1.1 Root Port Test

Topology: Platform Test Topology, PTC in Platform Test mode

Section Notes:

1. Root Port (DUT) is the CONFIG_RD requester and PTC is the completer for that request in this test.

Initial Conditions:

1. Platform is up and running, with drivers for PTC loaded and functioning.
2. PTC is disarmed and no trigger conditions set up.

Procedure:

1. MACRO_PTC_PROGRAM (CORRUPT_RESERVED_FIELDS_ACK_DLLP, CONFIG_RD_REQ, 1) // PTC will use non-zero values in at least one reserved field of the ACK DLLP generated for this TLP.
2. MACRO_PTC_ARM ()
3. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_PTC (VENDOR_DEV_ID)
4. MACRO_PTC_STATUS (ACTION COUNT)
5. MACRO_PTC_CLEANUP ()
6. Verify that the DUT did not retransmit the same CONFIG_RD_REQ TLP. If the DUT meets these criteria, the DUT passes the test.
7. If the DUT did retransmit the CONFIG_RD_REQ TLP, log it as DUT's failure.

3.1.1.2 Endpoint Device Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Root Port is the CONFIG_RD requester and Endpoint (DUT) is the completer for that request in this test.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.
2. PTC is disarmed and no trigger conditions set up.
3. DUT is running default traffic (if any) – that is, no application started yet.

Procedure:

1. For Function 0 in DUT:
    a. Clear Device Status register and verify that none of the error status bits are set.
    b. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)

c. MACRO_PTC_PROGRAM (CORRUPT_RESERVED_FIELDS_ACK_DLLP, CONFIG_RD_COMPLETION, 1) // PTC will use non-zero values in at least one reserved field of the ACK DLLP generated for CONFIG_RD_COMPLETION TLP.

d. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.

e. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID

f. MACRO_PTC_DISARM ()

g. MACRO_READ_DATA_FROM_PTC ()

h. Verify that:

i. DUT did not retransmit the CONFIG_RD_COMPLETION TLP.

ii. Verify that the DUT did not set any error status bits in Device Status register.

i. If the DUT meets all the above criteria, the DUT passes the test for Function 0.

j. If the DUT retransmitted the TLP as above, or if any error status bits are set in Device Status register, log it as DUT's failure.

2. For (Function=1; Function=7; Function++) in DUT:

a. Clear Device Status register and verify that none of the error status bits are set.

b. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)

c. MACRO_PTC_PROGRAM (CORRUPT_RESERVED_FIELDS_ACK_DLLP, ANY_TLP, 1) // ANY_TLP could be Config_Rd_Completion with Completion Status of either Successful Completion or Unsupported Request depending on whether the Function exists.

d. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.

e. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID)

f. MACRO_PTC_DISARM ()

g. MACRO_READ_DATA_FROM_PTC ()

h. Verify that:

i. DUT did not retransmit the CONFIG_RD_COMPLETION TLP.

ii. Verify that the DUT did not set any error status bits in Device Status register other than Unsupported Request Detected (bit 3) or Correctable Error Detected (bit 0).

i. If the DUT meets above criteria, the DUT passes the test.

j. If the DUT retransmitted the TLP as above, or if any error status bits are set in Device Status register, log it as DUT's failure.

3. If the DUT fails for any Function, treat the overall result as DUT's failure.

3.1.1.3 Switch and Bridge Downstream Port Test

Topology: Switch Test Topology, PTC in Platform Test mode

Section notes:

1. Algorithm same as in the Root Port Test except the DUT is a Switch's or Bridge's downstream port.

3.1.1.4 Switch and Bridge Upstream Port Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Algorithm same as in the Endpoint Device Test case except the DUT is the Switch's or Bridge's upstream port.

3.2 LCRC and Sequence Number Rules (TLP Transmitter)

3.2.1 Test 52-10 RetransmitOnNak

Test Introduction

The intent of this test is to ensure that a DUT will retransmit a transaction for which a NAK DLLP has been returned.

Section Notes:

1. Test applies to all PCI Express port types.
2. DATA_BUF – holds the data read back from the device.

3.2.1.1 Root Port Test

Topology: Platform Test Topology, PTC in Platform Test mode

Section Notes:

1. Root Port (DUT) is the CONFIG_RD requester and PTC is the completer for that request.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.
2. PTC is disarmed and no trigger conditions set up.

Procedure:

1. MACRO_PTC_PROGRAM (NAK, CONFIG_RD_REQ, 1)

2. MACRO_PTC_ARM ()
3. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_PTC (VENDOR_DEV_ID)
4. MACRO_PTC_STATUS (ACTION COUNT)
5. MACRO_PTC_CLEANUP ()
6. Verify that the DUT retransmits the CONFIG_RD_REQ TLP for which it received the NAK DLLP. If it did, the DUT passes the test.
7. If the DUT did not retransmit the CONFIG_RD_REQ TLP as above, log it as the DUT's failure.

3.2.1.2 Endpoint Device Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Root Port is the CONFIG_RD requester and Endpoint (DUT) is the completer for that request.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.
2. PTC is disarmed and no trigger conditions set up.
3. DUT is running default traffic (if any) – that is, no application started yet.

Procedure:

1. For Function 0 in DUT:
    a. Clear Device Status register and verify that none of the error status bits are set.
    b. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)
    c. MACRO_PTC_PROGRAM (NAK, CONFIG_RD_COMPLETION, 1)
    d. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.
    e. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID)
    f. MACRO_PTC_DISARM ()
    g. MACRO_READ_DATA_FROM_PTC ()
    h. Verify that:
        i. CONFIG_RD_COMPLETION TLP for which a NAK DLLP is received is retransmitted by the DUT.
        ii. Verify that the DUT did not set any error status bits in Device Status register.

i. If all of the conditions in the above step are met, DUT passes the test for Function 0. Otherwise consider it as DUT's failure.

2. For (Function=1; Function=7; Function++) in DUT:

a. Clear Device Status register and verify that none of the error status bits are set.

b. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)

c. MACRO_PTC_PROGRAM (NAK, ANY_TLP, 1) // ANY_ TLP could be Config_Rd_Completion with Completion Status of either Successful Completion or Unsupported Request depending on whether the Function exists.

d. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.

e. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID)

f. MACRO_PTC_DISARM ()

g. MACRO_READ_DATA_FROM_PTC ()

h. Verify that:

i. CONFIG_RD_COMPLETION TLP for which a NAK DLLP is received, is retransmitted by the DUT.

ii. Verify that the DUT did not set any other error status bits in Device Status register other than Unsupported Request Detected (bit 3) or Correctable Error Detected (bit 0).

i. If all of the conditions in the above step are met, DUT passes the test. Otherwise consider it as DUT's failure.

3. If the DUT fails for any Function, treat the overall result as DUT's failure.

3.2.1.3 Switch and Bridge Downstream Port Test

Topology: Switch test topology, PTC in platform test mode

Section Notes:

1. Algorithm same as in the Root Port Test except the DUT is a Switch's or Bridge's downstream port.

3.2.1.4 Switch and Bridge Upstream Port / Bridge Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Algorithm same as in the Endpoint device test case except the DUT is the Switch's or Bridge's upstream port.

3.2.2 Test 52-11 ReplayTimerTest

Test Introduction

The intent of this test is to ensure that a DUT's REPLAY_TIMER will cause it to retransmit a transaction when it does not receive either an ACK or a NAK. Also, it verifies that a correctable error message is controlled by the enable and mask bits.

Notes:

1. Test applies to all PCI Express port types.
2. DATA_BUF – holds the data read back from the device.

3.2.2.1 Root Port Test

Topology: Platform Test Topology, PTC in Platform Test mode

Section Notes:

1. Root Port (DUT) is the CONFIG_RD requester and the PTC is the completer for that request.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.
2. PTC is disarmed and no trigger conditions set up.

Procedure:

1. MACRO_PTC_PROGRAM (NO_ACK_NAK, CONFIG_RD_REQ, 1)
2. MACRO_PTC_ARM ()
3. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_PTC (VENDOR_DEV_ID)
4. MACRO_PTC_STATUS (ACTION COUNT)
5. MACRO_PTC_CLEANUP ()
6. Verify that the DUT retransmits the CONFIG_RD_REQ TLP for which it received no ACK or NAK DLLP. If it did, the DUT passes the test.
7. If the DUT did not retransmit the CONFIG_RD_REQ TLP as above, log it as DUT's failure.

3.2.2.2 Endpoint Device Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Root Port is the CONFIG_RD requester and Endpoint (DUT) is the completer for that request in this test.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.

2. PTC is disarmed and no trigger conditions set up.

3. DUT is running default traffic (if any) – that is, no application started yet.

Procedure:

1. For Function 0 in DUT:

    a. Clear Device Status register and verify that none of the error status bits are set.

    b. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)

    c. MACRO_PTC_PROGRAM (NO_ACK_NAK, CONFIG_RD_COMPLETION, 1)

    d. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.

    e. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID

    f. MACRO_PTC_DISARM ()

    g. MACRO_READ_DATA_FROM_PTC ()

    h. Verify that:

        i. Only Correctable Error bit (bit 0) in the DUT's Device Status register is set and DUT did not set any other error status bits in Device Status register.

        ii. CONFIG_RD_COMPLETION TLP is retransmitted by the DUT.

        iii. If error reporting is enabled in DUT's Device Control register (bit 0), ERR_COR message is sent by DUT.

        iv. If correctable error reporting is not enabled in DUT's Device Control register (bit 0), no ERR_COR message is sent.

    i. If all of the conditions in the above step are met, DUT passes the test for Function 0. Otherwise consider it as DUT's failure.

2. For (Function=1; Function=7; Function++) in DUT:

    a. Clear Device Status register and verify that none of the error status bits are set.

    b. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)

    c. MACRO_PTC_PROGRAM (NO_ACK_NAK, ANY_TLP, 1)//ANY_TLP could be Config_Rd_Completion with Completion Status of either Successful Completion or Unsupported Request depending on whether the Function exists.

    d. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.

    e. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID)

    f. MACRO_PTC_DISARM ()

g. MACRO_READ_DATA_FROM_PTC ()

h. Verify that:

i. Correctable Error bit (bit 0) in the DUT's Device Status register is set. Depending on whether the function exists or not the Unsupported Request bit (bit 3) in the DUT's Device Status register is allowed to be set, but the DUT did not set any other error status bits in Device Status register.

ii. CONFIG_RD_COMPLETION TLP is retransmitted by the DUT.

iii. If correctable error reporting is enabled in DUT's Device Control register (bit 0), ERR_COR message is sent by DUT.

iv. If correctable error reporting is not enabled in DUT's Device Control register (bit 0), no ERR_COR message is sent.

i. If all of the conditions in the above step are met, DUT passes the test. Otherwise consider it as DUT's failure.

3. If the DUT fails for any Function, treat the overall result as DUT's failure.

3.2.2.3 Switch and Bridge Downstream Port Test

Topology: Switch Test Topology, PTC in Platform Test mode

Section Notes:

1. Algorithm same as in the Root Port Test except the DUT is a Switch's or Bridge's downstream port.

3.2.2.4 Switch and Bridge Upstream / Bridge Port Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Algorithm same as in the Endpoint device test case except the DUT is the Switch's or Bridge's upstream port.

3.2.3 Test 52-12 ReplayNumTest

Test Introduction

The intent of this test is to ensure that a DUT will keep retransmitting a transaction for which a NAK DLLP has been issued until the number of times its REPLAY_NUM supports.

Notes:

1. Test applies to all PCI Express port types.
2. REPLAY_NUM – holds the maximum number of times a DUT can retransmit a TLP without causing link retraining.
3. DATA_PATTERN – 0xa5b4c3d2 (arbitrarily chosen).

4. BYTE_COUNT – number of bytes written into the PTC memory with the specified pattern – use value of one (forces a single TLP in all cases that can be NAKed).
5. DATA_BUF – holds the data read back from the device.

3.2.3.1 Root Port Test

Topology: Platform Test Topology, PTC in Platform Test mode

Section Notes:

1. Root Port (DUT) is the CONFIG_RD requester and the PTC is the completer for that request.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.
2. PTC is disarmed and no trigger conditions set up.

Procedure:

1. Set REPLAY_NUM = 3.
2. MACRO_PTC_PROGRAM (NAK, CONFIG_RD_REQ, REPLAY_NUM)
3. MACRO_PTC_ARM ()
4. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_PTC (VENDOR_DEV_ID)
5. MACRO_PTC_STATUS (ACTION COUNT)
6. MACRO_PTC_CLEANUP ()
7. Verify that the DUT retransmits the CONFIG_RD_REQ TLP for REPLAY_NUM of times with the same sequence number. If it did, the DUT passes the test.
8. If the DUT did not retransmit the TLP as above, log it as DUT's failure.

3.2.3.2 Endpoint Device Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode.

Section Notes:

1. Root Port is the CONFIG_RD requester and the Endpoint (DUT) is the completer for that request.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.
2. PTC is disarmed and no trigger conditions set up.
3. DUT is running default traffic (if any) – that is, no application started yet.

Procedure:

1. For Function 0 in DUT:

a. Clear Device Status register and verify that none of the error status bits are set.

b. Set REPLAY_NUM = 3.

c. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)

d. MACRO_PTC_PROGRAM (NAK, CONFIG_RD_COMPLETION, REPLAY_NUM)

e. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.

f. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID)

g. MACRO_PTC_DISARM ()

h. MACRO_READ_DATA_FROM_PTC ()

i. Verify that the DUT retransmits the CONFIG_RD_COMPLETION for REPLAY_NUM of times with the same sequence number and no error status bits are set in the Device Status register.

j. If the DUT meets above criteria, the DUT passes the test for Function 0. Otherwise consider it as DUT's failure.

2. For (Function=1; Function=7; Function++) in DUT:

a. Clear Device Status register and verify that none of the error status bits are set.

b. Set REPLAY_NUM = 3.

c. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)

d. MACRO_PTC_PROGRAM (NAK, ANY_TLP, REPLAY_NUM)//ANY_TLP could be Config_Rd_Completion with Completion Status of either Successful Completion or Unsupported Request depending on whether the Function exists.

e. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.

f. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID)

g. MACRO_PTC_DISARM ()

h. MACRO_READ_DATA_FROM_PTC ()

i. Verify that the DUT retransmits the CONFIG_RD_COMPLETION for REPLAY_NUM of times with the same sequence number. Verify that the DUT did not set any other error status bits other than Unsupported Request Detected bit (bit 3) or Correctable Error Detected (bit 0) depending on whether function exists or not.

j. If the DUT meets above criteria, the DUT passes the test. Otherwise consider it as DUT's failure.

3. If the DUT fails for any Function, treat the overall result as DUT's failure.

3.2.3.3 Switch and Bridge Downstream Port Test

Topology: Switch Test Topology, PTC in Platform Test mode

Section Notes:

1. Algorithm same as in the Root Port Test except the DUT is a Switch's or Bridge's downstream port.

3.2.3.4 Switch and Bridge Upstream Port / Bridge Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Algorithm same as in the Endpoint device test case except the DUT is the Switch's or Bridge's upstream port.

3.2.4 Test 52-20 LinkRetrainOnRetryFail

Test Introduction

The intent of this test is to ensure that the link connected to the DUT will go into retraining after trying for REPLAY_NUM of times to get a TLP acknowledged and failing because it receives NAK DLLPs instead. It will also test that while in retraining the retry buffer and link states are not changed and that the TLP in the retry buffer is retransmitted after link retraining completes. Also, it verifies that a REPLAY NUM Rollover error is logged. Finally, it verifies that a correctable error message is controlled by the enable and mask bits.

Section Notes:

1. Test applies to all PCI Express port types.
2. REPLAY_NUM – holds the maximum number of times a DUT can retransmit a TLP without causing link retraining. Use one more than this to cause retraining of link while still keeping Physical LinkUp = 1.
3. DATA_BUF – holds the data read back from the device.

3.2.4.1 Root Port Test

Topology: Platform Test Topology, PTC in Platform Test mode

Section Notes:

1. Root Port (DUT) is the MEM_RD requester and PTC is the completer for that request.
2. This memory read request target is the memory behind the PTC allocated through the BAR mechanism.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.

2. PTC is disarmed and no trigger conditions set up.

Procedure:

1. Set REPLAY_NUM = 3.
2. MACRO_PTC_PROGRAM (NAK, CONFIG_RD_REQ, REPLAY_NUM+1)
3. MACRO_PTC_ARM ()
4. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_PTC (VENDOR_DEV_ID)
5. MACRO_POLL_DUT_FOR_LINK_RETRAINING () // of the root port Link Status register connected to the PTC.
6. Verify that link got successfully retrained (confirm via a protocol analyzer if needed). If not, treat it as DUT failure.
7. MACRO_PTC_STATUS (POLL) // verify the action count is zero.
8. MACRO_PTC_CLEANUP ()
9. Verify that the link did not change its state. That is, the link should be in active state while undergoing link retraining (by checking to see that that the TS1 and TS2 sequences keep the lane and link numbers intact while retraining, and UpdateFC DLLPs are exchanged instead of InitFCx DLLPs). If not, treat it as DUT's failure.
10. Verify that DATA_BUF contains the Vendor and Device IDs of the PTC, written after the link got retrained. If not, treat it as DUT's failure.

3.2.4.2 Endpoint Device Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Root Port is the CONFIG_RD requester and the Endpoint (DUT) is the completer for that request in this test.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.
2. PTC is disarmed and no trigger conditions set up.
3. DUT is running default traffic (if any) – that is, no application started yet.

Procedure:

1. For Function 0 in DUT:
    a. Clear Device Status register and verify that none of the error status bits are set.
    b. Set REPLAY_NUM = 3.
    c. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)

d. MACRO_PTC_PROGRAM (NAK, CONFIG_RD_COMPLETION, REPLAY_NUM+1)

e. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.

f. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID)

g. MACRO_POLL_PTC_FOR_LINK_RETRAINING ()//of the port connected to the DUT.

h. MACRO_PTC_DISARM ()

i. MACRO_READ_DATA_FROM_PTC ()

j. Verify that:

CASE 1: PCIe1.0a, PCIe1.1, PCIe2.x, PCIe3.x - Correctable Error - AER Implemented

i. Link got successfully retrained through Link Status register of the PTC (confirm via a protocol analyzer if needed), an error is logged in the DUT's port register for REPLAY_NUM overflow, and that an error message is generated by the DUT. If not, treat it as DUT failure.

ii. Verify that DUT retransmitted Configuration data after the link retraining. If not, treat it as DUT failure.

iii. Only Correctable Error bit (bit 0) in the DUT's Device Status register is set and DUT did not set any other error status bits in Device Status register.

iv. If the REPLAY_NUM Rollover error is not masked in the Correctable Error Mask register (bit 8) of AER:

1) REPLAY_NUM Rollover status bit (bit 8) in the Correctable Error Status register of the AER is set.

2) If correctable error reporting is enabled in DUT's Device Control register (bit 0), ERR_COR message is sent by DUT.

3) If correctable error reporting is not enabled in DUT's Device Control register (bit 0), no ERR_COR message is sent.

v. If the REPLAY_NUM Rollover error is masked in the Correctable Error Mask register (bit 8) of AER, then no ERR_COR message is sent.

vi. If all of the conditions in the above steps are met, DUT passes the test for Function 0. Otherwise consider it as DUT's failure.

CASE 2: PCIe1.0a, PCIe1.1, PCIe2.x, PCIe3.x -Correctable Error – No AER Implemented

i. Link got successfully retrained through Link Status register of the PTC (confirm via a protocol analyzer if needed), an error is logged in the DUT's port register for

REPLAY_NUM overflow and that an error message is generated by the DUT. If not, treat it as DUT failure.

ii. Verify that DUT retransmitted Configuration data after the link retraining. If not, treat it as DUT failure.

iii. Only Correctable Error bit (bit 0) in the DUT's Device Status register is set and DUT did not set any other error status bits in Device Status register.

iv. If correctable error reporting is enabled in DUT's Device Control register (bit 0), ERR_COR message is sent by DUT.

v. If correctable error reporting is not enabled in DUT's Device Control register (bit 0), no ERR_COR message is sent.

vi. If all of the conditions in the above steps are met, DUT passes the test for Function 0. Otherwise consider it as DUT's failure.

2. For (Function=1; Function=7; Function++) in DUT:

a. Clear Device Status register and verify that none of the error status bits are set.

b. Set REPLAY_NUM = 3.

c. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)

d. MACRO_PTC_PROGRAM (NAK, ANY_TLP, REPLAY_NUM+1) //ANY_TLP could be Config_Rd_Completion with Completion Status of either Successful Completion or Unsupported Request depending on whether the Function exists.

e. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.

f. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID)

g. MACRO_POLL_PTC_FOR_LINK_RETRAINING ()//of the port connected to the DUT.

h. MACRO_PTC_DISARM ()

i. MACRO_READ_DATA_FROM_PTC ()

j. Verify that:

CASE 1: PCIe1.0a, PCIe1.1, PCIe2.x, PCIe3.x - Correctable Error - AER Implemented

i. Link got successfully retrained through Link Status register of the PTC (confirm via a protocol analyzer if needed), an error is logged in the DUT's port register for REPLAY_NUM overflow and that an error message is generated by the DUT. If not, treat it as DUT failure.

ii. Verify that DUT retransmitted Configuration data or Unsupported Request (depending on whether the function exists or not) after the link retraining. If not, treat it as DUT failure.

iii. Correctable Error bit (bit 0) in the DUT's Device Status register is set. Depending on whether the function exists or not the Unsupported Request bit (bit 3) in the DUT's Device Status register is allowed to be set but the DUT did not set any other error status bits in Device Status register.

iv. If the REPLAY_NUM Rollover error is not masked in the Correctable Error Mask register (bit 8) of AER:

1) REPLAY_NUM Rollover status bit (bit 8) in the Correctable Error Status register of the AER is set.

2) If correctable error reporting is enabled in DUT's Device Control register (bit 0), ERR_COR message is sent by DUT.

3) If correctable error reporting is not enabled in DUT's Device Control register (bit 0), no ERR_COR message is sent.

v. If the REPLAY_NUM Rollover error is masked in the Correctable Error Mask register (bit 8) of AER, then no ERR_COR message is sent.

vi. If all of the conditions in the above step are met, DUT passes the test. Otherwise consider it as DUT's failure.

CASE 2: PCIe1.0a, PCIe1.1, PCIe2.x, PCIe3.x -Correctable Error – No AER Implemented

i. Link got successfully retrained through the Link Status register of the PTC (confirm via a protocol analyzer if needed), an error is logged in the DUT's port register for REPLAY_NUM overflow, and that an error message is generated by the DUT. If not, treat it as DUT failure.

ii. Verify that retransmitted Configuration data or Unsupported Request (depending on whether the function exists or not) after the link retraining. If not, treat it as DUT failure.

iii. Correctable Error bit (bit 0) in the DUT's Device Status register is set. Depending on whether the function exists or not, the Unsupported Request bit (bit 3) in the DUT's Device Status register is allowed to be set but the DUT did not set any other error status bits in Device Status register.

iv. If correctable error reporting is enabled in DUT's Device Control register (bit 0), ERR_COR message is sent by DUT.

v. If correctable error reporting is not enabled in DUT's Device Control register (bit 0), no ERR_COR message is sent.

vi. If all of the conditions in the above step are met, DUT passes the test. Otherwise consider it as DUT's failure.

vii. If the DUT fails for any Function, treat the overall result as DUT's failure.

3.2.4.3 Switch and Bridge Downstream Port Test

Topology: Switch Test Topology, PTC in Platform Test mode

Section Notes:

1. Algorithm same as in the Root Port Test except the DUT is a Switch's or Bridge's downstream port.

3.2.4.4 Switch and Bridge Upstream Port / Bridge Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Algorithm same as in the Endpoint device test case except the DUT is the Switch's or Bridge's upstream port.

3.2.5 Test 52-100 ReplayTLPOrder

Test Introduction

The intent of this test is to verify that the oldest unacknowledged TLP is retransmitted first in replay followed by the other unacknowledged TLPs in the same order that they were transmitted first. Also, it verifies that a correctable error message is controlled by the enable and mask bits.

Notes:

1. Test applies to all PCI Express port types.
2. REPLAY_NUM – holds the maximum number of times a DUT can retransmit a TLP without causing link retraining. Use one more than this to cause retraining of link while still keeping Physical LinkUp = 1.
3. DATA_BUF – holds the data read back from the device.

3.2.5.1 Root Port Test

Topology: Platform Test Topology, PTC in Platform Test mode

Section Notes:

1. Root Port (DUT) is the requester and the PTC is the completer for any request in this test.
2. This memory read request target is the memory behind the PTC allocated through BAR mechanisms.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.
2. PTC is disarmed and no trigger conditions set up.

Procedure:

1. Set REPLAY_NUM = 3.
2. MACRO_PTC_PROGRAM (NO_ACK_NAK, CONFIG_RD_REQ, REPLAY_NUM)
3. MACRO_PTC_ARM ()
4. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_PTC (PCI_COMPATIBLE) // first 24 bytes in DWORD fashion.
5. MACRO_PTC_STATUS (ACTION COUNT)
6. MACRO_PTC_CLEANUP ()
7. Verify that the trace contains at least one or more replays of multiple TLPs, sent in the original order (based on TLP Sequence Number). If the replays satisfy the order condition, the DUT passes the test.
8. If the replay order does not satisfy the order condition, treat it as DUT's failure.

3.2.5.2 Endpoint Device Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Root Port is the CONFIG_RD requester and the Endpoint (DUT) is the completer for that request in this test.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.
2. PTC is disarmed and no trigger conditions set up.
3. DUT is running default traffic (if any) – that is, no application started yet.
4. Link training identified in its InitFCs that the Endpoint supports two or more NP credits. If that is not the case, the test is skipped with warning that the test could not be executed.

Procedure:

1. For Function 0 in DUT:
   a. Clear Device Status register and verify that none of the error status bits are set.
   b. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)
   c. MACRO_PTC_PROGRAM (NO_ACK_NAK, CONFIG_RD_COMPLETION, 1)
   d. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.
   e. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID)

f. MACRO_PTC_DISARM ()

g. MACRO_READ_DATA_FROM_PTC ()

h. Verify that:

i. DUT retransmitted CONFIG_RD_COMPLETION TLPs.

ii. Only Correctable Error bit (bit 0) in the DUT's Device Status register is set and DUT did not set any other error status bits in Device Status register.

iii. If correctable error reporting is enabled in DUT's Device Control register (bit 0), send ERR_COR message.

iv. If correctable error reporting is not enabled in DUT's Device Control register (bit 0), no ERR_COR message is sent.

i. If all of the conditions in the above step are met, DUT passes the test for Function 0. Otherwise consider it as DUT's failure.

2. For (Function=1; Function=7; Function++) in DUT:

a. Clear Device Status register and verify that none of the error status bits are set.

b. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)

c. MACRO_PTC_PROGRAM (NO_ACK_NAK, ANY_TLP, 1) // ANY_TLP could be Config_Rd_Completion with Completion Status of either Successful Completion or Unsupported Request depending on whether the Function exists.

d. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.

e. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID)

f. MACRO_PTC_DISARM ()

g. MACRO_READ_DATA_FROM_PTC ()

h. Verify that:

i. DUT retransmitted CONFIG_RD_COMPLETION TLPs.

ii. Only Correctable Error bit (bit 0) or Unsupported Request bit (bit 3) in the DUT's Device Status register is set and DUT did not set any other error status bits in Device Status register.

iii. If correctable error reporting is enabled in DUT's Device Control register (bit 0), send ERR_COR message.

iv. If correctable error reporting is not enabled in DUT's Device Control register (bit 0), no ERR_COR message is sent.

i. If all of the conditions in the above step are met, DUT passes the test. Otherwise consider it as DUT's failure.

3. If the DUT fails for any Function, treat the overall result as DUT's failure.

3.2.5.3 Switch and Bridge Downstream Port Test

Topology: Switch Test Topology, PTC in Platform Test mode

Section Notes:

1. Algorithm same as in the Root Port Test except the DUT is a Switch's or Bridge's downstream port.

3.2.5.4 Switch and Bridge Upstream Port / Bridge Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Algorithm same as in the Endpoint device test case except the DUT is the Switch's or Bridge's upstream port.

3.2.6 Test 52-150 CorruptedDLLP

Test Introduction

The intent of this test is to ensure that a DUT recognizes a DLLP with a bad CRC, drops it, and logs a Bad DLLP error. Also, it verifies that a correctable error message is controlled by the enable and mask bits.

Notes:

1. Test applies to all PCI Express port types.
2. DATA_BUF – holds the data read back from the device.

3.2.6.1 Root Port Test

Topology: Platform Test Topology, PTC in Platform Test mode

Section Notes:

1. Root Port (DUT) is the CONFIG_RD requester and the PTC is the completer for that request.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.
2. PTC is disarmed and no trigger conditions set up.

Procedure:

1. MACRO_PTC_PROGRAM (CORRUPT_ACK_CRC, CONFIG_RD_REQ, 1)
2. MACRO_PTC_ARM ()

3. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_PTC (VENDOR_DEV_ID)
4. MACRO_PTC_STATUS (ACTION COUNT)
5. MACRO_PTC_CLEANUP ()
6. Verify that the DUT retransmits the CONFIG_RD_REQ TLP for which it has received an ACK DLLP with a bad CRC and logs a Bad DLLP error associated with that port. If it did, the DUT passes the test.
7. If the DUT did not retransmit the TLP as above, log it as DUT's failure.

3.2.6.2 Endpoint Device Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Root Port is the CONFIG_RD requester and Endpoint (DUT) is the completer for that request.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.
2. PTC is disarmed and no trigger conditions set up.
3. DUT is running default traffic (if any) – that is, no application started yet.

Procedure:

1. For Function 0 in DUT:
   a. Clear Device Status register and verify that none of the error status bits are set.
   b. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)
   c. MACRO_PTC_PROGRAM (CORRUPT_ACK_CRC, CONFIG_RD_COMPLETION, 1)
   d. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.
   e. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID)
   f. MACRO_PTC_DISARM ()
   g. MACRO_READ_DATA_FROM_PTC ()
   h. Verify that:
      i. CASE 1: PCIe1.0a, PCIe1.1, PCIe2.x, PCIe3.x - Correctable Error - AER Implemented
         1) The CONFIG_RD_COMPLETION TLP for which the DUT received an ACK DLLP with a bad CRC is retransmitted by the DUT.

2) Only Correctable Error bit (bit 0) in the DUT's Device Status register is set and DUT did not set any other error status bits in Device Status register.

3) If the error is not masked in the Correctable Error Mask register (bit 7) of AER:

   a) Bad DLLP status bit (bit 7) in the Correctable Error Status register of the AER is set.

   b) If correctable error reporting is enabled in DUT's Device Control register (bit 0), ERR_COR message is sent by DUT.

   c) If correctable error reporting is not enabled in DUT's Device Control register (bit 0), no ERR_COR message is sent.

4) If the error is masked in the Correctable Error Mask register (bit 7) of AER, then no ERR_COR message is sent.

5) If all the above conditions are met, DUT passes the test for Function 0. Otherwise consider it as DUT's failure.

ii. CASE 2: PCIe1.0a, PCIe1.1, PCIe2.x, PCIe3.x - Correctable Error – No AER Implemented

1) The CONFIG_RD_COMPLETION TLP for which the DUT received an ACK DLLP with a bad CRC is retransmitted by the DUT.

2) Only Correctable Error bit (bit 0) in the DUT's Device Status register is set and DUT did not set any other error status bits in Device Status register.

3) If correctable error reporting is enabled in DUT's Device Control register (bit 0), ERR_COR message is sent by DUT.

4) If correctable error reporting is not enabled in DUT's Device Control register (bit 0), no ERR_COR message is sent.

5) If all the above conditions are met, DUT passes the test for Function 0. Otherwise consider it as DUT's failure.

2. For (Function=7; Function=7; Function++) in DUT:

   a. Clear Device Status register and verify that none of the error status bits are set.

   b. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)

   c. MACRO_PTC_PROGRAM (CORRUPT_ACK_CRC, ANY_TLP, 1) // ANY_TLP could be Config_Rd_Completion with Completion Status of either Successful Completion or Unsupported Request depending on whether the Function exists.

   d. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.

   e. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID)

f. MACRO_PTC_DISARM ()

g. MACRO_READ_DATA_FROM_PTC ()

h. Verify that:

i. CASE 1: PCIe1.0a, PCIe1.1, PCIe2.x, PCIe3.x - Correctable Error - AER Implemented

1) The CONFIG_RD_COMPLETION TLP for which the DUT received an ACK DLLP with a bad CRC is retransmitted by the DUT.

2) Correctable Error bit (bit 0) in the DUT's Device Status register is set. Depending on whether the function exists or not the Unsupported Request bit (bit 3) in the DUT's Device Status register is allowed to be set but the DUT did not set any other error status bits in Device Status register.

3) If the error is not masked in the Correctable Error Mask register (bit 7) of AER:

   a) Bad DLLP status bit (bit 7) in the Correctable Error Status register of the AER is set.

   b) If correctable error reporting is enabled in DUT's Device Control register (bit 0), ERR_COR message is sent by DUT.

   c) If correctable error reporting is not enabled in DUT's Device Control register (bit 0), no ERR_COR message is sent.

4) If the error is masked in the Correctable Error Mask register (bit 7) of AER, then no ERR_COR message is sent.

5) If all the above conditions are met, DUT passes the test. Otherwise consider it as DUT's failure.

ii. CASE 2: PCIe1.0a, PCIe1.1, PCIe2.x, PCIe3.x - Correctable Error – No AER Implemented

1) The CONFIG_RD_COMPLETION TLP for which the DUT received an ACK DLLP with a bad CRC is retransmitted by the DUT.

2) Correctable Error bit (bit 0) in the DUT's Device Status register is set. Depending on whether the function exists or not the Unsupported Request bit (bit 3) in the DUT's Device Status register is allowed to be set but the DUT did not set any other error status bits in Device Status register.

3) If correctable error reporting is enabled in DUT's Device Control register (bit 0), ERR_COR message is sent by DUT.

4) If correctable error reporting is not enabled in DUT's Device Control register (bit 0), no ERR_COR message is sent.

5) If all the above conditions are met, DUT passes the test. Otherwise consider it as DUT's failure.

3. If the DUT fails for any Function, treat the overall result as DUT's failure.

3.2.6.3 Switch and Bridge Downstream Port Test

Topology: Switch Test Topology, PTC in Platform Test mode

Section Notes:

1. Algorithm same as in the Root Port Test except the DUT is a Switch's or Bridge's downstream port.

3.2.6.4 Switch and Bridge Upstream Port / Bridge Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Algorithm same as in the Endpoint device test case except the DUT is the Switch's or Bridge's upstream port.

3.2.7 Test 52-160 UndefinedDLLPEncoding

Test Introduction

The intent of this test is to verify that the DUT silently drops any DLLP with undefined encoding (any pattern for the DLLP Type field that is reserved). Also, it verifies that a correctable error message is controlled by the enable bit.

Notes:

1. Test applies to all PCI Express port types.
2. DATA_BUF – holds the data read back from the device.

3.2.7.1 Root Port Test

Topology: Platform Test Topology, PTC in Platform Test mode

Section Notes:

1. Root Port (DUT) is the CONFIG_RD requester and the PTC is the completer for that request.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.
2. PTC is disarmed and no trigger conditions set up.

Procedure:

1. Read the DUT's advanced error reporting registers and save the values.
2. MACRO_PTC_PROGRAM (DLLP_UNDEFINED_ENCODING, CONFIG_RD_REQ, 1)
3. MACRO_PTC_ARM ()
4. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_PTC (VENDOR_DEV_ID)

5. MACRO_PTC_STATUS (ACTION COUNT)

6. MACRO_PTC_CLEANUP ()

7. Verify that the DUT silently drops the DLLP with undefined encoding by reading the DUT's advanced error reporting registers and verifying that they did not change from earlier read values (except for REPLAY_TIMER overflow being set). If they did not change, the DUT passes the test.

8. Verify that the DUT retries the configuration read after its ACK_NAK latency timer expires and that the transaction is completed successfully. If it did, the DUT passes the test.

9. If not, log it as DUT's failure.

3.2.7.2 Endpoint Device Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Root Port is the CONFIG_RD requester and Endpoint (DUT) is the completer for that request in this test.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.

2. PTC is disarmed and no trigger conditions set up.

3. DUT is running default traffic (if any) – that is, no application started yet.

Procedure:

1. For Function 0 in DUT;

   a. Clear Device Status register and verify that none of the error status bits are set.

   b. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)

   c. MACRO_PTC_PROGRAM (DLLP_UNDEFINED_ENCODING, CONFIG_RD_COMPLETION, 1)

   d. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.

   e. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID)

   f. MACRO_PTC_DISARM ()

   g. MACRO_READ_DATA_FROM_PTC ()

   h. Verify that:

      i. DUT retransmits the CONFIG_RD_COMPLETION TLP for which it received an undefined DLLP.

ii. Only Correctable Error bit (bit 0) in the DUT's Device Status register is set and DUT did not set any other error status bits in Device Status register.

iii. If correctable error reporting is enabled in DUT's Device Control register (bit 0), ERR_COR message is sent by DUT.

iv. If correctable error reporting is not enabled in DUT's Device Control register (bit 0), no ERR_COR message is sent.

i. If all of the conditions in the above step are met, DUT passes the test for function 0. Otherwise consider it as DUT's failure.

2. For (Function =1; Function=7; Function++) in DUT:

a. Clear Device Status register and verify that none of the error status bits are set.

b. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)

c. MACRO_PTC_PROGRAM (DLLP_UNDEFINED_ENCODING, ANY_TLP, 1) //ANY_TLP could be Config_Rd_Completion with Completion Status of either Successful Completion or Unsupported Request depending on whether the Function exists.

d. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.

e. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID)

f. MACRO_PTC_DISARM ()

g. MACRO_READ_DATA_FROM_PTC ()

h. Verify that:

i. DUT retransmits the CONFIG_RD_COMPLETION TLP for which it received an undefined DLLP.

ii. Correctable Error bit (bit 0) in the DUT's Device Status register is set. Depending on whether the function exists or not the Unsupported Request bit (bit 3) in the DUT's Device Status register is allowed to be set but the DUT did not set any other error status bits in Device Status register.

iii. If correctable error reporting is enabled in DUT's Device Control register (bit 0), ERR_COR message is sent by DUT.

iv. If correctable error reporting is not enabled in DUT's Device Control register (bit 0), no ERR_COR message is sent.

i. If all of the conditions in the above step are met, DUT passes the test. Otherwise consider it as DUT's failure.

3. If the DUT fails for any Function, treat the overall result as DUT's failure.

3.2.7.3 Switch and Bridge Downstream Port Test

Topology: Switch Test Topology, PTC in Platform Test mode

Section Notes:

1. Algorithm same as in the Root Port Test except the DUT is a Switch's or Bridge's downstream port.

3.2.7.4 Switch and Bridge Upstream Port / Bridge Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Algorithm same as in the Endpoint device test case except the DUT is the Switch's or Bridge's upstream port.

3.2.8 Test 52-170 WrongSeqNumInAckDLLP

Test Introduction

The intent of this test is to verify that the DUT drops any ACK DLLP that does not have a sequence number corresponding to an unacknowledged TLP and logs a Data Link Protocol error associated with the port. Also, it verifies that a correctable error message is controlled by the enable and mask bits.

Notes:

1. Test applies to all PCI Express port types.
2. DATA_BUF – holds the data read back from the device.

3.2.8.1 Root Port Test

Topology: Platform Test Topology, PTC in Platform Test mode

Section Notes:

1. Root Port (DUT) is the CONFIG_RD requester and the PTC is the completer for that request in this test.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.
2. PTC is disarmed and no trigger conditions set up.

Procedure:

1. MACRO_PTC_PROGRAM (ACK_DLLP_WRONG_SEQ_NUM, CONFIG_RD_REQ, 1) // PTC will supply an incorrect sequence number to the configuration read request TLP from RC.
2. MACRO_PTC_ARM ()
3. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_PTC (VENDOR_DEV_ID)

4. MACRO_PTC_STATUS (ACTION COUNT)
5. MACRO_PTC_CLEANUP ()
6. Verify that the DUT logs BAD DLLP error. If it did, the DUT passes the test.
7. If the DUT did not report an error, log it as DUT's failure.

3.2.8.2 Endpoint Device Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Root Port is the CONFIG_RD requester and Endpoint (DUT) is the completer for that request in this test.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.
2. PTC is disarmed and no trigger conditions set up.
3. DUT is running default traffic (if any) – that is, no application started yet.

Procedure:

1. For Function 0 in DUT:
   a. Clear Device Status register and verify that none of the error status bits are set.
   b. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)
   c. MACRO_PTC_PROGRAM (ACK_DLLP_WRONG_SEQ_NUM, CONFIG_RD_COMPLETION, 1) // PTC will issue an ACK with incorrect sequence number in response to the completion TLP.
   d. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.
   e. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID)
   f. MACRO_PTC_DISARM ()
   g. MACRO_READ_DATA_FROM_PTC ()
   h. Verify that:

      i. CASE 1: PCIe1.0a - Fatal Error - Severity – Non-Fatal - AER Implemented

      This implies DUT implemented AER and, in the Uncorrectable Severity register, Data Link Protocol Error Severity bit (bit 4) is cleared to indicate that this is not a fatal error.

      1) Only Non-Fatal Error bit (bit 1) in the DUT's Device Status register is set and DUT has not set any other error status bits in Device Status register.

a) If the error is not masked in the Uncorrectable Error Mask register (bit 4) of AER:

b) Data Link Protocol Error bit (bit 4) in the Uncorrectable Error Status register of the AER is set.

c) If non-fatal error reporting is enabled in DUT's Device Control register, send ERR_NONFATAL message.

2) If non-fatal error reporting is not enabled in DUT's Device Control register, no ERR_NONFATAL message is sent.

3) If the error is masked in the Uncorrectable Error Mask register (bit 4) of AER, then no ERR_NONFATAL message is sent.

4) If all the above conditions are met, DUT passes the test for Function 0. Otherwise consider it as DUT's failure.

ii. CASE 2: PCIe1.0a - Fatal Error - Severity – Fatal - AER Implemented

This implies DUT implemented AER and, in the Uncorrectable Severity register, Data Link Protocol Error Severity bit (bit 4) is set to indicate that this is a fatal error.

1) Only Fatal Error bit (bit 2) in the DUT's Device Status register is set and DUT has not set any other error status bits in Device Status register.

2) If the error is not masked in the Uncorrectable Error Mask register (bit 4) of AER:

a) Data Link Protocol Error bit (bit 4) in the Uncorrectable Error Status register of the AER is set.

b) If fatal error reporting is enabled in DUT's Device Control register, send ERR_FATAL message.

c) If fatal error reporting is not enabled in DUT's Device Control register, no ERR_FATAL message is sent.

3) If the error is masked in the Uncorrectable Error Mask register (bit 4) of AER, then no ERR_FATAL message is sent.

4) If all of the above conditions are met, DUT passes the test for Function 0. Otherwise consider it as DUT's failure.

iii. CASE 3: PCIe1.0a - Fatal Error - No AER

This implies DUT has no AER and the Malformed TLP is handled as a fatal error.

1) Only Fatal Error bit (bit 2) in the DUT's Device Status register is set and DUT has not set any other error status bits in Device Status register.

2) If fatal error reporting is enabled in DUT's Device Control register, send ERR_FATAL message.

3) If fatal error reporting is not enabled in DUT's Device Control register, no ERR_FATAL message is sent.

4) If all of the above conditions are met, DUT passes the test for Function 0. Otherwise consider it as DUT's failure.

iv. CASE 4: PCIe1.1 - Fatal Error - No AER
PCIe2.x - Fatal Error - No AER
PCIe3.x - Fatal Error - No AER

This implies DUT has no AER and the Malformed TLP is handled as a fatal error.

1) Only Fatal Error bit (bit 2) in the DUT's Device Status register is set and DUT has not set any other error status bits in Device Status register.

2) If fatal error reporting is enabled in DUT's Device Control register, send ERR_FATAL message.

3) If fatal error reporting is not enabled in DUT's Device Control register, no ERR_FATAL message is sent.

4) If all of the above conditions are met, DUT passes the test for Function 0. Otherwise consider it as DUT's failure.

v. CASE 5: PCIe1.1 – Fatal Error – Severity – Non-Advisory - AER Implemented
PCIe2.x – Fatal Error – Severity – Non-Advisory - AER Implemented
PCIe3.x – Fatal Error – Severity – Non-Advisory - AER Implemented

This implies DUT implemented AER and, in the Uncorrectable Severity register, Data Link Protocol Error Severity bit (bit 4) is set to indicate that this is a fatal error.

1) Only Fatal Error bit (bit 2) in the DUT's Device Status register is set and DUT has not set any other error status bits in Device Status register.

2) Data Link Protocol Error bit (bit 4) in the DUT's Uncorrectable Error Status register is set.

3) If the error is not masked in the Uncorrectable Error Mask register (bit 4) of AER:

   a) If fatal error reporting is enabled in DUT's Device Control register or SERR# Enable in Command register is set send ERR_FATAL message.

   b) If fatal error reporting is not enabled in DUT's Device Control register and SERR# Enable in Command register is not set, no ERR_FATAL message is sent.

4) If the error is masked in the Uncorrectable Error Mask register (bit 4) of AER, then no ERR_FATAL message is sent.

5) If all of the above conditions are met, then DUT passes the test for Function 0. Otherwise consider it as DUT's failure.

2. For (Function =1; Function=7; Function++) in DUT:

a. Clear Device Status register and verify that none of the error status bits are set.

b. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)

c. MACRO_PTC_PROGRAM (ACK_DLLP_WRONG_SEQ_NUM, ANY_TLP, 1)
//ANY_TLP could be Config_Rd_Completion with Completion Status of either Successful Completion or Unsupported Request depending on whether the Function exists.

d. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.

e. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID)

f. MACRO_PTC_DISARM ()

g. MACRO_READ_DATA_FROM_PTC ()

h. Verify that:

i. CASE 1: PCIe1.0a - Fatal Error - Severity – Non-Fatal - AER Implemented

This implies DUT implemented AER and, in the Uncorrectable Severity register, Data Link Protocol Error Severity bit (bit 4) is cleared to indicate that this is not a fatal error.

1) Unsupported Request is sent depending on whether the function exists or not.

2) Only Non-Fatal Error bit (bit 1) or Unsupported Request bit (bit 3) in the DUT's Device Status register is set and DUT has not set any other error status bits in Device Status register.

3) If the error is not masked in the Uncorrectable Error Mask register (bit 4) of AER:

a) Data Link Protocol Error bit (bit 4) in the Uncorrectable Error Status register of the AER is set.

b) If non-fatal error reporting is enabled in DUT's Device Control register, send ERR_NONFATAL message.

c) If non-fatal error reporting is not enabled in DUT's Device Control register, no ERR_NONFATAL message is sent.

4) If the error is masked in the Uncorrectable Error Mask register (bit 4) of AER, then no ERR_NONFATAL message is sent.

5) If all the above conditions are met, DUT passes the test. Otherwise consider it as DUT's failure.

ii. CASE 2: PCIe1.0a - Fatal Error - Severity – Fatal - AER Implemented

This implies DUT implemented AER and, in the Uncorrectable Severity register, Data Link Protocol Error Severity bit (bit 4) is set to indicate that this is a fatal error.

1) Unsupported Request is sent depending on whether the function exists or not.

2) Only Fatal Error bit (bit 2) or Unsupported Request bit (bit 3) in the DUT's Device Status register is set and DUT has not set any other error status bits in Device Status register.

3) If the error is not masked in the Uncorrectable Error Mask register (bit 4) of AER:

   a) Data Link Protocol Error bit (bit 4) in the Uncorrectable Error Status register of the AER is set.

   b) If fatal error reporting is enabled in DUT's Device Control register, send ERR_FATAL message.

   c) If fatal error reporting is not enabled in DUT's Device Control register, no ERR_FATAL message is sent.

4) If the error is masked in the Uncorrectable Error Mask register (bit 4) of AER, then no ERR_FATAL message is sent.

5) If all of the above conditions are met, DUT passes the test. Otherwise consider it as DUT's failure.

iii. CASE 3: PCIe1.0a - Fatal Error - No AER

This implies DUT has no AER and the Malformed TLP is handled as a fatal error.

1) Unsupported Request is sent depending on whether the function exists or not.

2) Only Fatal Error bit (bit 2) or Unsupported Request bit (bit 3) in the DUT's Device Status register is set and DUT has not set any other error status bits in Device Status register.

3) If fatal error reporting is enabled in DUT's Device Control register, send ERR_FATAL message.

4) If fatal error reporting is not enabled in DUT's Device Control register, no ERR_FATAL message is sent.

5) If all of the above conditions are met, DUT passes the test. Otherwise consider it as DUT's failure.

iv. CASE 4: PCIe1.1 - Fatal Error - No AER
PCIe2.x - Fatal Error - No AER
PCIe3.x - Fatal Error - No AER

This implies DUT has no AER and the Malformed TLP is handled as a fatal error.

1) Unsupported Request is sent depending on whether the function exists or not.

2) Only Fatal Error bit (bit 2) or Unsupported Request bit (bit 3) in the DUT's Device Status register is set and DUT has not set any other error status bits in Device Status register.

3) If fatal error reporting is enabled in DUT's Device Control register, send ERR_FATAL message.

4) If fatal error reporting is not enabled in DUT's Device Control register, no ERR_FATAL message is sent.

5) If all of the above conditions are met, DUT passes the test. Otherwise consider it as DUT's failure.

v. CASE 5: PCIe1.1 – Fatal Error – Severity – Non-Advisory - AER Implemented
PCIe2.x – Fatal Error – Severity – Non-Advisory - AER Implemented
PCIe3.x – Fatal Error – Severity – Non-Advisory - AER Implemented

This implies DUT implemented AER and, in the Uncorrectable Severity register, Data Link Protocol Error Severity bit (bit 4) is set to indicate that this is a fatal error.

1) Unsupported Request is sent depending on whether the function exists or not.

2) Only Fatal Error bit (bit 2) or Unsupported Request bit (bit 3) in the DUT's Device Status register is set and DUT has not set any other error status bits in Device Status register.

3) Data Link Protocol Error bit (bit 4) in the DUT's Uncorrectable Error Status register is set.

a) If the error is not masked in the Uncorrectable Error Mask register (bit 4) of AER:

b) If fatal error reporting is enabled in DUT's Device Control register or SERR# Enable in Command register is set send ERR_FATAL message.

4) If fatal error reporting is not enabled in DUT's Device Control register and SERR# Enable in Command register is not set, no ERR_FATAL message is sent.

5) If the error is masked in the Uncorrectable Error Mask register (bit 4) of AER, then no ERR_FATAL message is sent.

6) If all of the above conditions are met, then DUT passes the test. Otherwise consider it as DUT's failure.

3. If the DUT fails for any Function, treat the overall result as DUT's failure.

3.2.8.3 Switch and Bridge Downstream Port Test

Topology: Switch Test Topology, PTC in Platform Test mode

Section Notes:

1. Algorithm same as in the Root Port Test except the DUT is a Switch's or Bridge's downstream port.

3.2.8.4 Switch and Bridge Upstream Port / Bridge Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Algorithm same as in the Endpoint device test case except the DUT is the Switch's or Bridge's upstream port.

3.3 LCRC and Sequence Number (TLP Receiver)

3.3.1 Test 53-20 BadLCRC

Test Introduction

The intent of this test is to verify that a receiver discards a TLP with bad CRC by sending it a NAK and reporting a Bad TLP error associated with the port. Also, it verifies that a correctable error message is controlled by the enable and mask bits.

Notes:

1. Test applies to all PCI Express port types.
2. DATA_BUF – holds the data read back from the device.

3.3.1.1 Root Port Test

Topology: Platform Test Topology, PTC in Platform Test mode

Section Notes:

1. Root Port (DUT) is the requester and the PTC is the completer for that request in this test.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.
2. PTC is disarmed and no trigger conditions set up.

Procedure:

1. MACRO_PTC_PROGRAM (CORRUPT_LCRC, CONFIG_RD_COMPLETION, 1)
2. MACRO_PTC_ARM ()
3. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_PTC (VENDOR_DEV_ID)
4. MACRO_PTC_STATUS (ACTION COUNT)
5. MACRO_PTC_CLEANUP ()
6. Verify that DUT generated a NAK for the CONFIG_RD_COMPLETION TLP. If not, treat it as DUT's failure.

7. Verify that a Bad TLP port error was logged by the DUT for the port connected to the PTC. If not, treat it as DUT's failure.

3.3.1.2 Endpoint Device Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Root Port is the CONFIG_RD requester and the PTC is the forwarder of that request to the DUT (with bad LCRC).

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.
2. PTC is disarmed and no trigger conditions set up.
3. DUT is running default traffic (if any) – that is, no application started yet.

Procedure:

1. For Function 0 in DUT:
    a. Clear Device Status register and verify that none of the error status bits are set.
    b. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)
    c. MACRO_PTC_PROGRAM (CORRUPT_LCRC, CONFIG_RD_REQ, 1)
    d. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.
    e. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID)
    f. MACRO_PTC_DISARM ()
    g. MACRO_READ_DATA_FROM_PTC ()
    h. Verify that:
        i. CASE 1: PCIe1.0a, PCIe1.1, PCIe2.x, PCIe3.x - Correctable Error - AER Implemented
            1) DUT NAKed the TLP with bad CRC.
            2) Only Correctable Error bit (bit 0) in the DUT's Device Status register is set and DUT did not set any other error status bits in Device Status register.
            3) If the error is not masked in the Correctable Error Mask register (bit 6) of AER:
                a) Bad TLP status bit (bit 6) in the Correctable Error Status register of the AER is set.
                b) If correctable error reporting is enabled in DUT's Device Control register (bit 0), ERR_COR message is sent by DUT.

c) If correctable error reporting is not enabled in DUT's Device Control register (bit 0), no ERR_COR message is sent.

4) If the error is masked in the Correctable Error Mask register (bit 6) of AER, then no ERR_COR message is sent.

5) If all the above conditions are met, then DUT passes the test for Function 0. Otherwise consider it as DUT's failure.

ii. CASE 2: PCIe1.0a, PCIe1.1, PCIe2.x, PCIe3.x - Correctable Error – No AER Implemented

1) DUT NAKed the TLP with bad CRC.

2) Only Correctable Error bit (bit 0) in the DUT's Device Status register is set and DUT did not set any other error status bits in Device Status register.

3) If correctable error reporting is enabled in DUT's Device Control register (bit 0), ERR_COR message is sent by DUT.

4) If correctable error reporting is not enabled in DUT's Device Control register (bit 0), no ERR_COR message is sent.

5) If all the above conditions are met, then DUT passes the test for Function 0. Otherwise consider it as DUT's failure.

2. For (Function=1; Function=7; Function++) in DUT:

a. Clear Device Status register and verify that none of the error status bits are set.

b. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)

c. MACRO_PTC_PROGRAM (CORRUPT_LCRC, CONFIG_RD_REQ, 1)

d. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.

e. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID)

f. MACRO_PTC_DISARM ()

g. MACRO_READ_DATA_FROM_PTC ()

h. Verify that:

i. CASE 1: PCIe1.0a, PCIe1.1, PCIe2.x, PCIe3.x - Correctable Error - AER Implemented

1) DUT NAKed the TLP with bad CRC.

2) Correctable Error bit (bit 0) in the DUT's Device Status register is set. Depending on whether the function exists or not the Unsupported Request bit (bit 3) in the DUT's Device Status register is allowed to be set but the DUT did not set any other error status bits in Device Status register.

3) If the error is not masked in the Correctable Error Mask register (bit 6) of AER:

   a) Bad TLP status bit (bit 6) in the Correctable Error Status register of the AER is set.

   b) If correctable error reporting is enabled in DUT's Device Control register (bit 0), ERR_COR message is sent by DUT.

   c) If correctable error reporting is not enabled in DUT's Device Control register (bit 0), no ERR_COR message is sent.

4) If the error is masked in the Correctable Error Mask register (bit 6) of AER, then no ERR_COR message is sent.

5) If all the above conditions are met, then DUT passes the test. Otherwise consider it as DUT's failure.

ii. CASE 2: PCIe1.0a, PCIe1.1, PCIe2.x, PCIe3.x - Correctable Error – No AER Implemented

1) DUT NAKed the TLP with bad CRC.

2) Correctable Error bit (bit 0) in the DUT's Device Status register is set. Depending on whether the function exists or not the Unsupported Request bit (bit 3) in the DUT's Device Status register is allowed to be set but the DUT did not set any other error status bits in Device Status register.

3) If correctable error reporting is enabled in DUT's Device Control register (bit 0), ERR_COR message is sent by DUT.

4) If correctable error reporting is not enabled in DUT's Device Control register (bit 0), no ERR_COR message is sent.

5) If all the above conditions are met, then DUT passes the test. Otherwise consider it as DUT's failure.

3. If the DUT fails for any Function, treat the overall result as DUT's failure.

3.3.1.3 Switch and Bridge Downstream Port Test

Topology: Switch Test Topology, PTC in Platform Test mode

Section Notes:

1. Algorithm same as in the Root Port Test except the DUT is a Switch's or Bridge's downstream port.

3.3.1.4 SWITCH AND BRIDGE UPSTREAM PORT / BRIDGE TEST

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Algorithm same as in the Endpoint device test case except the DUT is the Switch's or Bridge's upstream port.

3.3.2 Test 53-31 DuplicateTLPSeqNum

Test Introduction

The intent of this test is to verify that duplicate TLPs (i.e., a TLP with the same sequence number associated at the link layer as that in the last 2048 TLPs received and acknowledged) are handled properly by the DUT, in that the duplicate will receive an acknowledge, but otherwise is discarded.

Notes:

1. Test applies to all PCI Express port types.
2. DATA_BUF – holds the data read back from the device.

3.3.2.1 Root Port Test

Topology: Platform Test Topology, PTC in Platform Test mode

Section Notes:

1. Root Port is the CONFIG_RD requester and the PTC is the completer for that request in this test.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.
2. PTC is disarmed and no trigger conditions set up.

Procedure:

1. MACRO_PTC_PROGRAM (DUPLICATE_TLP, CONFIG_RD_COMPLETION, 1)
2. MACRO_PTC_ARM ()
3. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_PTC (VENDOR_DEV_ID)
4. MACRO_PTC_STATUS (ACTION COUNT)
5. MACRO_PTC_CLEANUP ()
6. Verify that the Root Port transmits either a single coalesced ACK DLLP or two ACK DLLPs for the duplicate TLPs. If it did, the DUT passes the test.
7. If the DUT did not transmit either a coalesced ACK DLLP or a two ACK DLLPs as above, log it as DUT's failure.

3.3.2.2 Endpoint Device Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Root Port is the CONFIG_RD requester and DUT is the completer for that request.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.

2. PTC is disarmed and no trigger conditions set up.

3. DUT is running default traffic (if any) – that is, no application started yet.

Procedure:

1. For Function 0 in DUT:

    a. Clear Device Status register and verify that none of the error status bits are set.

    b. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)

    c. MACRO_PTC_PROGRAM (DUPLICATE_TLP, CONFIG_RD_REQ, 1)

    d. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.

    e. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID)

    f. MACRO_PTC_DISARM ()

    g. MACRO_READ_DATA_FROM_PTC ()

    h. Verify that:

        i. CONFIG_RD_REQ TLP which has been duplicated has received either two ACK DLLPs or a single coalesced ACK DLLP, but only a single completion is sent by the DUT.

        ii. Verify that the DUT did not set any error status bits in Device Status register.

    i. If the DUT meets above criteria, the DUT passes the test for the Function 0.

2. For (Function=1; Function=7; Function++) in DUT:

    a. Clear Device Status register and verify that none of the error status bits are set.

    b. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)

    c. MACRO_PTC_PROGRAM (DUPLICATE_TLP, CONFIG_RD_REQ, 1)

    d. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.

    e. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID)

    f. MACRO_PTC_DISARM ()

    g. MACRO_READ_DATA_FROM_PTC ()

    h. Verify that:

i. CONFIG_RD_REQ TLP which has been duplicated has received either two ACK DLLPs or a single coalesced ACK DLLP, but only a single completion is sent by the DUT.

ii. Verify that the DUT did not set any error status bits in Device Status register other than Unsupported Request Detected (bit 3) or Correctable Error Detected (bit 0), depending on whether the functions exist or not.

i. If the DUT meets above criteria, the DUT passes the test.

3. If the DUT fails for any Function, treat the overall result as DUT's failure.

3.3.2.3 Switch and Bridge Downstream Port Test

Topology: Switch Test Topology, PTC in Platform Test mode

Section Notes:

1. Algorithm same as in the Root Port Test except the DUT is a Switch's or Bridge's downstream port.

3.3.2.4 Switch and Bridge Upstream Port / Bridge Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Algorithm same as in the Endpoint device test case except the DUT is the Switch's or Bridge's upstream port.

3.4 Transaction Layer Rules

3.4.1 Test 54-12 TXN_BFT_RequestCompletion_UR

Test Introduction

The intent of this test is to verify that the DUT will issue a completion with Unsupported Request completion status for the configuration requests to function numbers that it does not support. In addition the DUT will send the appropriate error messages, depending on the support for role-based error reporting.

Note: At this point the algorithm applies to single function devices only. It may be expanded to handle multi-function devices in the future.

Notes:

1. Test applies to all PCI Express port types.

2. DATA_BUF – holds the data read back from the device.

3.4.1.1 Root Port Test

Topology: Platform Test Topology, PTC in Platform Test mode

Section Notes:

1. Root Port (DUT) is the CONFIG_RD requester and PTC is the completer for that request in this test.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.
2. PTC is disarmed and no trigger conditions set up.

Procedure:

Note: None at this time, but may be added later.

3.4.1.2 Endpoint Device Test

Topology: Endpoint Test Topology, PTC in Add-in Card Test mode

Section Notes:

1. Root Port is the CONFIG_RD requester and DUT is the completer for that request in this test.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.
2. PTC is disarmed and no trigger conditions set up.
3. DUT is running default traffic (if any) – that is, no application started yet.

Procedure:

1. Clear Device Status register by writing to the Device Control register of the DUT and verify that none of the error status bits are set.
2. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR)
3. MACRO_PTC_PROGRAM (DELAY_ACK_NAK_LEGAL, CFG_RD, 1) // just to kick start the trace buffer in the PTC and is a benign command to act on.
4. MACRO_PTC_ARM () // starts the trace buffer capture of all TLP headers from DUT.
5. For (Function=0; Function=7; Function++) in DUT:
    a. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID) // expectation is, not all eight functions are implemented in any DUT. In the case they are, the test will abort indicating as such and is not considered a valid test for that device.
6. MACRO_PTC_DISARM ()
7. MACRO_READ_DATA_FROM_PTC ()

CASE 1: PCIe1.0a - Non-Fatal Error - Severity – Non-Fatal - AER Implemented

This implies the DUT implemented AER and, in the Uncorrectable Severity register, Unsupported Request bit (bit 20) is cleared to indicate that this is not a fatal error.

1. Verify that:
   a. The DUT sent one or more completions with Unsupported Request completion status for all functions not implemented.
   b. Non-Fatal Error bit (bit 1) in the DUT's Device Status register is set.
   c. Unsupported Request bit (bit 3) in the DUT's Device Status register is set.
   d. If the error is not masked in the Uncorrectable Error Mask register (bit 20) of AER
      i. Unsupported Request bit (bit 20) in the Uncorrectable Error Status register of the AER is set.
      ii. If Unsupported Request error reporting is enabled in DUT's Device Control register, ERR_NONFATAL message is sent by DUT.
      iii. If Unsupported Request error reporting is not enabled in DUT's Device Control register, no ERR_NONFATAL message is sent.
   e. If the error is masked in the Uncorrectable Error Mask register (bit 20) of AER then no ERR_NONFATAL message is sent.
2. If all of the conditions in the above step are met, DUT passes the test. Otherwise consider it as DUT's failure.

CASE 2: PCIe1.0a - Non-Fatal Error - Severity – Fatal - AER Implemented

This implies DUT implemented AER is and in the Uncorrectable Severity register, Unsupported Request bit (bit 20) is set to indicate that this is a fatal error.

1. Verify that:
   a. The DUT sent one or more completions with Unsupported Request completion status for all functions not implemented.
   b. Fatal Error bit (bit 2) in the DUT's Device Status register is set.
   c. Unsupported Request bit (bit 3) in the DUT's Device Status register is set.
   d. Unsupported Request bit (bit 20) in the Uncorrectable Error Status register of the AER is set.
   e. If the error is not masked in the Uncorrectable Error Mask register (bit 20) of AER:
      i. If Unsupported Request error reporting is enabled in DUT's Device Control register, ERR_FATAL message is sent.

ii. If Unsupported Request error reporting is not enabled in DUT's Device Control register, no ERR_FATAL message is sent.

f. If the error is masked in the Uncorrectable Error Mask register (bit 20) of AER, then no ERR_FATAL message is sent.

2. If all of the conditions in the above step are met, DUT passes the test. Otherwise consider it as DUT's failure.

CASE 3: PCIe1.0a - Non-Fatal Error - No AER

This implies DUT has no AER and the Unsupported Request is handled as a non-fatal error.

1. Verify that:

a. The DUT sent one or more completions with Unsupported Request completion status for all functions not implemented.

b. Non-Fatal Error bit (bit 1) in the DUT's Device Status register is set.

c. Unsupported Request bit (bit 3) in the DUT's Device Status register is set.

d. If Unsupported Request error reporting is enabled in DUT's Device Control register, send ERR_NONFATAL message.

i. If Unsupported Request error reporting is not enabled in DUT's Device Control register, no ERR_NONFATAL message is sent

2. If all of the conditions in the above step are met, DUT passes the test. Otherwise consider it as DUT's failure.

CASE 4: PCIe1.1, PCIe2.x, PCIe3.x - Non-Fatal Error - Severity – Advisory - AER Implemented

This implies the DUT implemented AER and, in the Uncorrectable Severity register, Unsupported Request bit (bit 20) is cleared to indicate that this is not a fatal error.

1. Verify that:

a. The DUT sent one or more completions with Unsupported Request completion status for all functions not implemented.

b. Correctable Error bit (bit 0) in the DUT's Device Status register is set and Non-Fatal Error bit (bit 1) and Fatal Error bit (bit 2) are clear.

c. Unsupported Request bit (bit 3) in the DUT's Device Status register is set.

d. Advisory Non-Fatal bit (bit 13) in the Correctable Error register of the AER is set.

e. If the error is not masked in the Correctable Error Mask register (bit 13) of AER:

i. Unsupported Request bit (bit 20) in the Uncorrectable Error Status register of the AER is set.

ii. If Unsupported Request error reporting is enabled in DUT's Device Control register, send ERR_COR message.

iii. If Unsupported Request error reporting is not enabled in DUT's Device Control register, no ERR_COR message is sent.

f. If the error is masked in the Correctable Error Mask register (bit 13) of AER, then no ERR_COR message is sent.

2. If all of the conditions in the above step are met, DUT passes the test. Otherwise consider it as DUT's failure.

CASE 5: PCIe1.1, PCIe2.x, PCIe3.x - Non-Fatal Error – No AER Implemented

This implies DUT has no AER and the Unsupported Request is handled as a non-fatal error.

1. Verify that:

a. The DUT sent one or more completions with Unsupported Request completion status for all functions not implemented.

b. Non-Fatal Error bit (bit 1) in the DUT's Device Status register is set.

c. Unsupported Request bit (bit 3) in the DUT's Device Status register is set.

d. No ERR_NONFATAL message is sent.

2. If all of the conditions in the above bullet are met, DUT passes the test. Otherwise consider it as DUT's failure.

CASE 6: PCIe1.1, PCIe2.x, PCIe3.x – Fatal Error – Severity – Non-Advisory - AER Implemented

This implies the DUT implemented AER and, in the Uncorrectable Severity register, Unsupported Request bit (bit 20) is set to indicate that this is a fatal error.

1. Verify that:

a. The DUT sent one or more completions with Unsupported Request completion status for all functions not implemented.

b. Fatal Error bit (bit 2) in the DUT's Device Status register is set and Non-Fatal Error bit (bit 1) and Correctable Error bit (bit 0) are clear.

c. Unsupported Request bit (bit 3) in the DUT's Device Status register is set.

d. Unsupported Request bit (bit 20) in the DUT's Uncorrectable Error Status register is set.

e. If the error is not masked in the Uncorrectable Error Mask register (bit 20) of AER:

i. If Unsupported Request error reporting is enabled in DUT's Device Control register or SERR# Enable in Command register is set, ERR_FATAL message is sent by DUT.

ii. If Unsupported Request error reporting is not enabled in DUT's Device Control register or SERR# Enable in Command register is not set, no ERR_FATAL message is sent.

f. If the error is masked in the Uncorrectable Error Mask register (bit 20) of AER, then no ERR_FATAL message is sent

2. If all of the conditions in the above step are met, DUT passes the test. Otherwise consider it as DUT's failure.

3.4.1.3 Switch and Bridge Downstream Port Test

Topology: Switch Test Topology, PTC in Platform Test mode

Section Notes:

1. Algorithm same as in the RC test case except the DUT is a Switch's or Bridge's downstream port.

3.4.1.4 Switch and Bridge Upstream Port Test

Topology: Endpoint Test Topology, PTC in Add-in Card Test mode

Section Notes:

2. Algorithm same as in the Endpoint device test case except the DUT is the Switch's or Bridge's upstream port.

3.4.2 Test 54-20 BadECRC

Test Introduction

The intent of this test is to verify that the DUT that is not enabled to generate ECRC will not generate a TLP Digest in a transmitted TLP. It also verifies that the DUT that is enabled to generate ECRC will generate a TLP Digest with the correct ECRC in a transmitted TLP. It also verifies that the DUT that is not enabled to check ECRC ignores the TLP Digest in a received TLP that targets the DUT. It also verifies that the DUT that is enabled to check ECRC will check the TLP Digest (if present) in a received TLP that targets the DUT and if the ECRC is incorrect it logs an ECRC error associated with the port. Finally, it verifies that an uncorrectable error message is controlled by the enable, mask, and severity bits.

Notes:

1. Test applies to all PCI Express device and port types that have a link.

2. DATA_BUF – holds the data read back from the device.

3.4.2.1 Root Port Test

Topology: Platform Test Topology, PTC in Platform Test mode

Section notes:

1. Root Port (DUT) is the CONFIG_RD requester and the PTC is the completer for that request in this test.

Initial Conditions:

1. Platform is up and running, with drivers for the PTC loaded and functioning.
2. PTC is disarmed and no trigger conditions set up.

Procedure:

Test Case 1 Only:

1. MACRO_PTC_PROGRAM (GENERATE_ECRC, CONFIG_RD_COMPLETION, 1) // PTC will add a TLP digest with a valid ECRC to the next CONFIG READ COMPLETION it transmits.

Test Case 2 Only:

1. MACRO_PTC_PROGRAM (CORRUPT_ECRC, CONFIG_RD_COMPLETION, 1) // PTC will add a TLP digest with an invalid ECRC to the next CONFIG READ COMPLETION it transmits.

All Test Cases:

1. MACRO_PTC_ARM ()
2. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_PTC (VENDOR_DEV_ID)
3. MACRO_PTC_STATUS (ACTION COUNT)
4. MACRO_PTC_CLEANUP ()
5. Verify that:
    a. The DUT generated an ACK DLLP for the CONFIG_RD_COMPLETION TLP, regardless of the ECRC value.
6. If all of the conditions above are met the DUT passes the test.
7. If any of the conditions above are not met, log it as DUT's failure.

3.4.2.2 Endpoint Device Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Root Port is the CONFIG_RD requester and Endpoint (DUT) is the completer for that request in this test.

Initial Conditions:

1. If AER_IMPLEMENTED_FLAG=0, then the DUT does not support ECRC generation or ECRC checking, so set ECRC_GEN_IMPLEMENTED_FLAG=0 and ECRC_CHECK_IMPLEMENTED_FLAG=0.

2. If AER_IMPLEMENTED_FLAG=1, then software determines if the DUT supports ECRC generation, by checking the ECRC Generation Capable bit in the Advanced Error Capabilities and Control register. If the bit returns 1 then the DUT's port supports ECRC generation, so set ECRC_GEN_IMPLEMENTED_FLAG=1. If the bit returns 0, then the DUT's port does not support ECRC generation, so set ECRC_GEN_IMPLEMENTED_FLAG=0.

3. If AER_IMPLEMENTED_FLAG=1, then software determines if the DUT supports ECRC checking, by checking the ECRC Check Capable bit in the Advanced Error Capabilities and Control register. If the bit returns 1 then the DUT's port supports ECRC checking, so set ECRC_CHECK_IMPLEMENTED_FLAG=1. If the bit returns 0, then the DUT's port does not support ECRC checking, so set ECRC_CHECK_IMPLEMENTED_FLAG=0.

4. Platform is up and running, with drivers for the PTC loaded and functioning.

5. PTC is disarmed and no trigger conditions set up.

6. DUT is running default traffic (if any) – that is no application started yet.

7. DUT has received at least one configuration write with the correct value (i.e., Bus Number and Device Number) that the DUT will use as its Requester ID for all TLPs it generates.

PROCEDURE:

1. For Function Number=0 in DUT:

   a. If AER_IMPLEMENTED_FLAG=1, then clear all the bits in the DUT's Uncorrectable Error Status register and verify that the ECRC Error Status bit is not set.

   b. Clear Correctable Error Detected, Non-Fatal Error Detected, Fatal Error Detected, Unsupported Request Detected bits (Device Status register) and verify that none of these error status bits are set.

   c. Program the following values to the indicated registers in the DUT:

      TEST CASE 1: SERR# Enable = disabled, Non-Fatal Error Reporting = enabled, ECRC Error = unmasked, ECRC Error = fatal severity, ECRC Error = not checked

      i. If ECRC_CHECK_IMPLEMENTED_FLAG=1, then clear the ECRC Check Enable bit (Advanced Error Capabilities and Control register) to 0.

      ii. Clear SERR# Enable bit (Command register) to 0.

      iii. Set Non-Fatal Error Reporting Enable bit (Device Control register) to 1.

      iv. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Mask bit (Uncorrectable Error Mask register) to 0.

      v. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Severity bit (Uncorrectable Error Mask register) to 0.

TEST CASE 2: (Only if ECRC_CHECK_IMPLEMENTED_FLAG=1) SERR# Enable = disabled, Non-Fatal Error Reporting = enabled, ECRC Error = unmasked, ECRC Error = non-fatal severity, ECRC Error = no error

i. If ECRC_CHECK_IMPLEMENTED_FLAG=1, then set the ECRC Check Enable bit (Advanced Error Capabilities and Control register) to 1.

ii. Clear SERR# Enable bit (Command register) to 0.

iii. Set Non-Fatal Error Reporting Enable bit (Device Control register) to 1.

iv. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Mask bit (Uncorrectable Error Mask register) to 0.

v. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Severity bit (Uncorrectable Error Mask register) to 0.

TEST CASE 3: (Only if ECRC_CHECK_IMPLEMENTED_FLAG=1) SERR# Enable = disabled, Non-Fatal Error Reporting = enabled, ECRC Error = unmasked, ECRC Error = non-fatal severity

i. If ECRC_CHECK_IMPLEMENTED_FLAG=1, then set the ECRC Check Enable bit (Advanced Error Capabilities and Control register) to 1.

ii. Clear SERR# Enable bit (Command register) to 0.

iii. Set Non-Fatal Error Reporting Enable bit (Device Control register) to 1.

iv. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Mask bit (Uncorrectable Error Mask register) to 0.

v. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Severity bit (Uncorrectable Error Mask register) to 0.

TEST CASE 4: (Only if ECRC_CHECK_IMPLEMENTED_FLAG=1) SERR# Enable = enabled, Non-Fatal Error Reporting = disabled, ECRC Error = unmasked, ECRC Error = non-fatal severity

i. If ECRC_CHECK_IMPLEMENTED_FLAG=1, then set the ECRC Check Enable bit (Advanced Error Capabilities and Control register) to 1.

ii. Set SERR# Enable bit (Command register) to 1.

iii. Clear Non-Fatal Error Reporting Enable bit (Device Control register) to 0.

iv. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Mask bit (Uncorrectable Error Mask register) to 0.

v. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Severity bit (Uncorrectable Error Mask register) to 0.

TEST CASE 5: (Only if ECRC_CHECK_IMPLEMENTED_FLAG=1) SERR# Enable = disabled, Non-Fatal Error Reporting = disabled, ECRC Error = unmasked, ECRC Error = non-fatal severity

i. If ECRC_CHECK_IMPLEMENTED_FLAG=1, then set the ECRC Check Enable bit (Advanced Error Capabilities and Control register) to 1.

ii. Clear SERR# Enable bit (Command register) to 0.

iii. Clear Non-Fatal Error Reporting Enable bit (Device Control register) to 0.

iv. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Mask bit (Uncorrectable Error Mask register) to 0.

v. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Severity bit (Uncorrectable Error Mask register) to 0.

TEST CASE 6: (Only if ECRC_CHECK_IMPLEMENTED_FLAG=1) SERR# Enable = disabled, Non-Fatal Error Reporting = enabled, ECRC Error = masked, ECRC Error = non-fatal severity

i. If ECRC_CHECK_IMPLEMENTED_FLAG=1, then set the ECRC Check Enable bit (Advanced Error Capabilities and Control register) to 1.

ii. Clear SERR# Enable bit (Command register) to 0.

iii. Set Non-Fatal Error Reporting Enable bit (Device Control register) to 1.

iv. If AER_IMPLEMENTED_FLAG=1, then set the ECRC Error Mask bit (Uncorrectable Error Mask register) to 1.

v. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Severity bit (Uncorrectable Error Mask register) to 0.

TEST CASE 7: (Only if ECRC_CHECK_IMPLEMENTED_FLAG=1) SERR# Enable = disabled, Fatal Error Reporting = enabled, ECRC Error = unmasked, ECRC Error = fatal severity

i. If ECRC_CHECK_IMPLEMENTED_FLAG=1, then set the ECRC Check Enable bit (Advanced Error Capabilities and Control register) to 1.

ii. Clear SERR# Enable bit (Command register) to 0.

iii. Set Fatal Error Reporting Enable bit (Device Control register) to 1.

iv. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Mask bit (Uncorrectable Error Mask register) to 0.

v. If AER_IMPLEMENTED_FLAG=1, then set the ECRC Error Severity bit (Uncorrectable Error Mask register) to 1.

TEST CASE 8: (Only if ECRC_CHECK_IMPLEMENTED_FLAG=1) SERR# Enable = enabled, Fatal Error Reporting = disabled, ECRC Error = unmasked, ECRC Error = fatal severity

i. If ECRC_CHECK_IMPLEMENTED_FLAG=1, then set the ECRC Check Enable bit (Advanced Error Capabilities and Control register) to 1.

ii. Set SERR# Enable bit (Command register) to 1.

iii. Clear Fatal Error Reporting Enable bit (Device Control register) to 0.

iv. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Mask bit (Uncorrectable Error Mask register) to 0.

v. If AER_IMPLEMENTED_FLAG=1, then set the ECRC Error Severity bit (Uncorrectable Error Mask register) to 1.

TEST CASE 9: (Only if ECRC_CHECK_IMPLEMENTED_FLAG=1) SERR# Enable = disabled, Fatal Error Reporting = disabled, ECRC Error = unmasked, ECRC Error = fatal severity

i. If ECRC_CHECK_IMPLEMENTED_FLAG=1, then set the ECRC Check Enable bit (Advanced Error Capabilities and Control register) to 1.

ii. Clear SERR# Enable bit (Command register) to 0.

iii. Clear Fatal Error Reporting Enable bit (Device Control register) to 0.

iv. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Mask bit (Uncorrectable Error Mask register) to 0.

v. If AER_IMPLEMENTED_FLAG=1, then set the ECRC Error Severity bit (Uncorrectable Error Mask register) to 1.

TEST CASE 10: (Only if ECRC_CHECK_IMPLEMENTED_FLAG=1) SERR# Enable = disabled, Fatal Error Reporting = enabled, ECRC Error = masked, ECRC Error = fatal severity

i. If ECRC_CHECK_IMPLEMENTED_FLAG=1, then set the ECRC Check Enable bit (Advanced Error Capabilities and Control register) to 1.

ii. Clear SERR# Enable bit (Command register) to 0.

iii. Set Fatal Error Reporting Enable bit (Device Control register) to 1.

iv. If AER_IMPLEMENTED_FLAG=1, then set the ECRC Error Mask bit (Uncorrectable Error Mask register) to 1.

v. If AER_IMPLEMENTED_FLAG=1, then set the ECRC Error Severity bit (Uncorrectable Error Mask register) to 1.

ALL TEST CASES:

d. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR) // trace buffer to capture all TLP headers from DUT.

TEST CASE 2 ONLY:

e. MACRO_PTC_PROGRAM (GENERATE_ECRC, CONFIG_RD_REQ, 1) // PTC will add a TLP digest with a valid ECRC to the next CONFIG READ it transmits.

TEST CASE 1 and TEST CASES 3 to 10 ONLY:

f. MACRO_PTC_PROGRAM (CORRUPT_ECRC, CONFIG_RD_REQ, 1) // PTC will add a TLP digest with an invalid ECRC to the next CONFIG READ it transmits.

ALL TEST CASES:

g. MACRO_PTC_ARM () // starts the trace buffer capture.

h. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID)

i. MACRO_PTC_DISARM () // ends the trace buffer capture.

j. MACRO_READ_DATA_FROM_PTC () // get the trace buffer.

k. Verify that:

TEST CASE 1 ONLY:

i. No ERR_NONFATAL Message TLP with this Function's Function Number in the Requester ID field is sent by DUT.

ii. The DUT did not set any of the four error status bits in the Device Status register.

iii. If AER_IMPLEMENTED_FLAG=1, then ECRC Error Status bit in the DUT's Uncorrectable Error Status register is not set.

TEST CASE 2 ONLY:

i. No ERR_NONFATAL Message TLP is sent by DUT.

ii. The DUT did not set any of the four error status bits in the Device Status register.

iii. If AER_IMPLEMENTED_FLAG=1, then ECRC Error Status bit in the DUT's Uncorrectable Error Status register is not set.

TEST CASE 3, TEST CASE 4 ONLY:

i. DUT transmits ERR_NONFATAL Message TLP using TC=0x0.

ii. The Non-Fatal Error Detected bit in the DUT's Device Status register is set and DUT did not set any of the other three error status bits in the Device Status register.

iii. If AER_IMPLEMENTED_FLAG=1, then ECRC Error Status bit in the DUT's Uncorrectable Error Status register is set.

TEST CASE 5, TEST CASE 6 ONLY:

i. No ERR_NONFATAL Message TLP with this Function's Function Number in the Requester ID field is sent by DUT.

ii. The Non-Fatal Error Detected bit in the DUT's Device Status register is set and DUT did not set any of the other three error status bits in the Device Status register

iii. If AER_IMPLEMENTED_FLAG=1, then ECRC Error Status bit in the DUT's Uncorrectable Error Status register is set.

TEST CASE 7, TEST CASE 8 ONLY:

i. DUT transmits ERR_FATAL Message TLP using TC=0x0.

ii. The Fatal Error Detected bit in the DUT's Device Status register is set and DUT did not set any of the other three error status bits in the Device Status register.

iii. If AER_IMPLEMENTED_FLAG=1, then ECRC Error Status bit in the DUT's Uncorrectable Error Status register is set.

TEST CASE 9, TEST CASE 10 ONLY:

i. No ERR_ FATAL Message TLP with this Function's Function Number in the Requester ID field is sent by DUT.

ii. The Fatal Error Detected bit in the DUT's Device Status register is set and DUT did not set any of the other three error status bits in the Device Status register.

iii. If AER_IMPLEMENTED_FLAG=1, then ECRC Error Status bit in the DUT's Uncorrectable Error Status register is set.

ALL TEST CASES:

l. If all of the conditions above are met then DUT passes the test for Function 0.

m. If any of the conditions above are not met, log it as DUT's failure.

2. For (Function Number =1; Function Number=7; Function Number ++) in DUT:

   a. Clear Correctable Error Detected, Non-Fatal Error Detected, Fatal Error Detected, Unsupported Request Detected bits (Device Status register).

   b. Check the Unsupported Request Detected bit (Device Status register):

i. If the Unsupported Request Detected bit is set, then this Function Number does not correspond to an implemented function in the DUT, so set the software flag FUNCTION_EXISTS_FLAG=0.

ii. If the Unsupported Request Detected bit is clear, then this Function Number correspond to an implemented function in the DUT, so set the software flag FUNCTION_EXISTS_FLAG=1.

c. If FUNCTION_EXISTS_FLAG is 1 and if AER_IMPLEMENTED_FLAG=1, then clear all the bits in the DUT's Uncorrectable Error Status register and verify that the ECRC Error Status bit is not set.

d. If FUNCTION_EXISTS_FLAG is 1, clear Correctable Error Detected, Non-Fatal Error Detected, Fatal Error Detected, Unsupported Request Detected bits (Device Status register) and verify that none of the four error status bits in the Device Status register are set.

e. If FUNCTION_EXISTS_FLAG is 1, program the following values to the indicated registers in the DUT:

TEST CASE 1: SERR# Enable = disabled, Non-Fatal Error Reporting = enabled, ECRC Error = unmasked, ECRC Error = non-fatal severity, ECRC Error = not checked

i. If ECRC_CHECK_IMPLEMENTED_FLAG=1, then clear the ECRC Check Enable bit (Advanced Error Capabilities and Control register) to 0.

ii. Clear SERR# Enable bit (Command register) to 0.

iii. Set Non-Fatal Error Reporting Enable bit (Device Control register) to 1.

iv. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Mask bit (Uncorrectable Error Mask register) to 0.

v. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Severity bit (Uncorrectable Error Mask register) to 0.

TEST CASE 2: (Only if ECRC_CHECK_IMPLEMENTED_FLAG=1) SERR# Enable = disabled, Non-Fatal Error Reporting = enabled, ECRC Error = unmasked, ECRC Error = non-fatal severity, ECRC Error = no error

i. If ECRC_CHECK_IMPLEMENTED_FLAG=1, then clear the ECRC Check Enable bit (Advanced Error Capabilities and Control register) to 1.

ii. Clear SERR# Enable bit (Command register) to 0.

iii. Set Non-Fatal Error Reporting Enable bit (Device Control register) to 1.

iv. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Mask bit (Uncorrectable Error Mask register) to 0.

v. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Severity bit (Uncorrectable Error Mask register) to 0.

TEST CASE 3: (Only if ECRC_CHECK_IMPLEMENTED_FLAG=1) SERR# Enable = disabled, Non-Fatal Error Reporting = enabled, ECRC Error = unmasked, ECRC Error = non-fatal severity

i. If ECRC_CHECK_IMPLEMENTED_FLAG=1, then clear the ECRC Check Enable bit (Advanced Error Capabilities and Control register) to 1.

ii. Clear SERR# Enable bit (Command register) to 0.

iii. Set Non-Fatal Error Reporting Enable bit (Device Control register) to 1.

iv. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Mask bit (Uncorrectable Error Mask register) to 0.

v. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Severity bit (Uncorrectable Error Mask register) to 0.

TEST CASE 4: (Only if ECRC_CHECK_IMPLEMENTED_FLAG=1) SERR# Enable = enabled, Non-Fatal Error Reporting = disabled, ECRC Error = unmasked, ECRC Error = non-fatal severity

i. If ECRC_CHECK_IMPLEMENTED_FLAG=1, then clear the ECRC Check Enable bit (Advanced Error Capabilities and Control register) to 1.

ii. Set SERR# Enable bit (Command register) to 1.

iii. Clear Non-Fatal Error Reporting Enable bit (Device Control register) to 0.

iv. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Mask bit (Uncorrectable Error Mask register) to 0.

v. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Severity bit (Uncorrectable Error Mask register) to 0.

TEST CASE 5: (Only if ECRC_CHECK_IMPLEMENTED_FLAG=1) SERR# Enable = disabled, Fatal Error Reporting = disabled, ECRC Error = unmasked, ECRC Error = non-fatal severity

i. If ECRC_CHECK_IMPLEMENTED_FLAG=1, then clear the ECRC Check Enable bit (Advanced Error Capabilities and Control register) to 1.

ii. Clear SERR# Enable bit (Command register) to 0.

iii. Clear Fatal Error Reporting Enable bit (Device Control register) to 0.

iv. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Mask bit (Uncorrectable Error Mask register) to 0.

v. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Severity bit (Uncorrectable Error Mask register) to 0.

TEST CASE 6: (Only if ECRC_CHECK_IMPLEMENTED_FLAG=1) SERR# Enable = disabled, Non-Fatal Error Reporting = enabled, ECRC Error = masked, ECRC Error = non-fatal severity

i. If ECRC_CHECK_IMPLEMENTED_FLAG=1, then clear the ECRC Check Enable bit (Advanced Error Capabilities and Control register) to 1.

ii. Clear SERR# Enable bit (Command register) to 0.

iii. Set Non-Fatal Error Reporting Enable bit (Device Control register) to 1.

iv. If AER_IMPLEMENTED_FLAG=1, then set the ECRC Error Mask bit (Uncorrectable Error Mask register) to 1.

v. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Severity bit (Uncorrectable Error Mask register) to 0.

TEST CASE 7: (Only if ECRC_CHECK_IMPLEMENTED_FLAG=1) SERR# Enable = disabled, Fatal Error Reporting = enabled, ECRC Error = unmasked, ECRC Error = fatal severity

i. If ECRC_CHECK_IMPLEMENTED_FLAG=1, then clear the ECRC Check Enable bit (Advanced Error Capabilities and Control register) to 1.

ii. Clear SERR# Enable bit (Command register) to 0.

iii. Set Fatal Error Reporting Enable bit (Device Control register) to 1.

iv. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Mask bit (Uncorrectable Error Mask register) to 0.

v. If AER_IMPLEMENTED_FLAG=1, then set the ECRC Error Severity bit (Uncorrectable Error Mask register) to 1.

TEST CASE 8: (Only if ECRC_CHECK_IMPLEMENTED_FLAG=1) SERR# Enable = enabled, Fatal Error Reporting = disabled, ECRC Error = unmasked, ECRC Error = fatal severity

i. If ECRC_CHECK_IMPLEMENTED_FLAG=1, then clear the ECRC Check Enable bit (Advanced Error Capabilities and Control register) to 1.

ii. Set SERR# Enable bit (Command register) to 1.

iii. Clear Fatal Error Reporting Enable bit (Device Control register) to 0.

iv. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Mask bit (Uncorrectable Error Mask register) to 0.

v. If AER_IMPLEMENTED_FLAG=1, then set the ECRC Error Severity bit (Uncorrectable Error Mask register) to 1.

TEST CASE 9: (Only if ECRC_CHECK_IMPLEMENTED_FLAG=1) SERR# Enable = disabled, Fatal Error Reporting = disabled, ECRC Error = unmasked, ECRC Error = fatal severity

i. If ECRC_CHECK_IMPLEMENTED_FLAG=1, then clear the ECRC Check Enable bit (Advanced Error Capabilities and Control register) to 1.

ii. Clear SERR# Enable bit (Command register) to 0.

iii. Clear Fatal Error Reporting Enable bit (Device Control register) to 0.

iv. If AER_IMPLEMENTED_FLAG=1, then clear the ECRC Error Mask bit (Uncorrectable Error Mask register) to 0.

v. If AER_IMPLEMENTED_FLAG=1, then set the ECRC Error Severity bit (Uncorrectable Error Mask register) to 1.

TEST CASE 10: (Only if ECRC_CHECK_IMPLEMENTED_FLAG=1) SERR# Enable = disabled, Fatal Error Reporting = enabled, ECRC Error = masked, ECRC Error = fatal severity

i. If ECRC_CHECK_IMPLEMENTED_FLAG=1, then clear the ECRC Check Enable bit (Advanced Error Capabilities and Control register) to 1.

ii. Clear SERR# Enable bit (Command register) to 0.

iii. Set Non-Fatal Error Reporting Enable bit (Device Control register) to 1.

iv. If AER_IMPLEMENTED_FLAG=1, then set the ECRC Error Mask bit (Uncorrectable Error Mask register) to 1.

v. If AER_IMPLEMENTED_FLAG=1, then set the ECRC Error Severity bit (Uncorrectable Error Mask register) to 1.

ALL TEST CASES:

f. MACRO_PTC_CONFIG_TRACE_BUF (TLP_HEADERS_ONLY, UPSTREAM_DIR) // trace buffer to capture all TLP headers from DUT.

TEST CASE 2 ONLY:

g. MACRO_PTC_PROGRAM (GENERATE_ECRC, CONFIG_RD_REQ, 1) // PTC will add a TLP digest with a valid ECRC to the next CONFIG READ it transmits.

TEST CASE 1 and TEST CASES 3 to 10 ONLY:

h. MACRO_PTC_PROGRAM (CORRUPT_ECRC, CONFIG_RD_REQ, 1) // PTC will add a TLP digest with an invalid ECRC to the next CONFIG READ it transmits.

ALL TEST CASES:

i. MACRO_PTC_ARM () // starts the trace buffer capture.

j. DATA_BUF = MACRO_READ_CONFIG_DATA_FROM_DUT (VENDOR_DEV_ID)

k. MACRO_PTC_DISARM () // ends the trace buffer capture.

l. MACRO_READ_DATA_FROM_PTC () // get the trace buffer.

m. Verify that:

TEST CASE 1 ONLY:

i. No ERR_NONFATAL Message TLP with this Function's Function Number in the Requester ID field is sent by DUT (other Message TLPs are ignored, as they are a side-effect of this test).

ii. If FUNCTION_EXISTS_FLAG is 1, the DUT did not set any of the DUT did not set any of the four error status bits in the Device Status register.

iii. If FUNCTION_EXISTS_FLAG is 1 and if AER_IMPLEMENTED_FLAG=1, then ECRC Error Status bit in the DUT's Uncorrectable Error Status register is not set.

TEST CASE 2 ONLY:

i. If FUNCTION_EXISTS_FLAG is 1, no ERR_NONFATAL Message TLP is sent by DUT.

ii. If FUNCTION_EXISTS_FLAG is 1, the DUT did not set any of the four error status bits in the Device Status register.

iii. If FUNCTION_EXISTS_FLAG is 1 and if AER_IMPLEMENTED_FLAG=1, then ECRC Error Status bit in the DUT's Uncorrectable Error Status register is not set.

TEST CASE 3, TEST CASE 4 ONLY:

i. If FUNCTION_EXISTS_FLAG is 1, DUT transmits ERR_NONFATAL Message TLP using TC=0x0.

ii. If FUNCTION_EXISTS_FLAG is 1, the Non-Fatal Error Detected bit in the DUT's Device Status register is set and DUT did not set any of the other three error status bits in the Device Status register.

iii. If FUNCTION_EXISTS_FLAG is 1 and if AER_IMPLEMENTED_FLAG=1, then ECRC Error Status bit in the DUT's Uncorrectable Error Status register is set.

TEST CASE 5, TEST CASE 6 ONLY:

i. If FUNCTION_EXISTS_FLAG is 1, No ERR_NONFATAL Message TLP with this Function's Function Number in the Requester ID field is sent by DUT.

ii. If FUNCTION_EXISTS_FLAG is 1, the Non-Fatal Error Detected bit in the DUT's Device Status register is set and DUT did not set any of the other three error status bits in the Device Status register.

iii. If FUNCTION_EXISTS_FLAG is 1 and if AER_IMPLEMENTED_FLAG=1, then ECRC Error Status bit in the DUT's Uncorrectable Error Status register is set.

TEST CASE 7, TEST CASE 8 ONLY:

i. If FUNCTION_EXISTS_FLAG is 1, DUT transmits ERR_FATAL Message TLP using TC=0x0.

ii. If FUNCTION_EXISTS_FLAG is 1, the Fatal Error Detected bit in the DUT's Device Status register is set and DUT did not set any of the other three error status bits in the Device Status register.

iii. If FUNCTION_EXISTS_FLAG is 1 and if AER_IMPLEMENTED_FLAG=1, then ECRC Error Status bit in the DUT's Uncorrectable Error Status register is set.

TEST CASE 9, TEST CASE 10 ONLY:

i. If FUNCTION_EXISTS_FLAG is 1, No ERR_FATAL Message TLP with this Function's Function Number in the Requester ID field is sent by DUT.

ii. If FUNCTION_EXISTS_FLAG is 1, the Fatal Error Detected bit in the DUT's Device Status register is set and DUT did not set any of the other three error status bits in the Device Status register.

iii. If FUNCTION_EXISTS_FLAG is 1 and if AER_IMPLEMENTED_FLAG=1, then ECRC Error Status bit in the DUT's Uncorrectable Error Status register is set.

ALL TEST CASES:

n. If all of the conditions above are met, then DUT passes the test for that Function.

o. If any of the conditions above are not met, log it as DUT's failure.

3. If the DUT fails for any Function, treat the overall result as DUT's failure.

3.4.2.3 Switch and Bridge Downstream Port Test

Topology: Switch Test Topology, PTC in Platform Test mode

SECTION NOTES:

1. Algorithm same as in the Root Port Test except the DUT is a Switch's or Bridge's downstream port.

3.4.2.4 Switch and Bridge Upstream Port Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

SECTION NOTES:

1. Algorithm same as in the Endpoint Device Test except the DUT is a Switch's or Bridge's upstream port.

3.5 Test 55-10 Reserved Bits in Training Sequences

The intent of these tests is to verify that the DUT will ignore the actual received value in the Reserved bits of the TS ordered Sets, and treat the Reserved bits as if they contained zero. The definition of which bits are Reserved depends on which Base specification level is being used:

For PCIe1.x:

a. Bits 0, 2:7 in symbol 4 (Data Rate Identifier)
b. Bits 4:7 in symbol 5 (Training Control)

For PCIe2.x:

c. Bits 0, 3:5 in symbol 4 (Data Rate Identifier)
d. Bits 5:7 in symbol 5 (Training Control)
a. Bit 4 in symbol 5 of TS2 (Training Control)

For PCIe3.x:

e. Bits 0, 4:5 in symbol 4 (Data Rate Identifier)
f. Bits 5:7 in symbol 5 (Training Control)
a. Bit 4 in symbol 5 of TS2 (Training Control)

3.5.1 Endpoint Device Test

Initial Conditions:

Platform and the DUT are powered, platform initiates fundamental reset to the DUT, and platform is up and running, with drivers for the test platform loaded and functioning. The link is in Detect state.

Note: Once all criteria for the platform LTSSM to move to the next LTSSM state have been met (as specified in Chapter 4 of the PCI Express Base Specification), the platform must make the transition to the next state within 500 μs.

1. Platform initiates link training into L0 state, advertising 2.5 GT/s data rate support. For this training all the bits in the TS1/TS2 that are reserved in the appropriate Base Specification are set

to 0 in every TS1/TS2 transmitted. Verify that link training is successful by checking that Idle Symbols are transmitted by the DUT at 2.5 GT/s, or by checking that the link is operational at 2.5 GT/s upon entering L0. To check that the link is operational, platform will send InitFCx DLLPs to complete flow control initialization, send a configuration read to register 0 (vendor / device ID), and verify that valid data is returned by the DUT. Otherwise, log as DUT failure and note LTSSM state at which link training is stuck.

2. Platform brings link into Detect state by initiating fundamental reset to the DUT.

3. Platform initiates link training into L0 state advertising 2.5 GT/s data rate support. For this training all the bits in the TS1/TS2 that are reserved in the appropriate Base Specification are set to 1 in every TS1/TS2 transmitted.

   Verify that link training is successful by checking that Idle Symbols are transmitted by the DUT at 2.5 GT/s, or by checking that the link is operational at 2.5 GT/s upon entering L0. To check that the link is operational, platform will send InitFCx DLLPs to complete flow control initialization, send a configuration read to register 0 (vendor / device ID), and verify that valid data is returned by the DUT. Otherwise, log as DUT failure and note LTSSM state at which link training is stuck.

4. Steps 2 and 3 are repeated; this time platform initiates link retraining by bringing the link to Recovery.RcvrLock, to Recovery.RcvrCfg, to Recovery.Idle, and back to L0 state. During Recovery state the specified reserved bits in step 3 are still set to 1 for every TS1/TS2 transmitted. Verify that link retraining is successful by checking that Idle Symbols are transmitted by the DUT at 2.5 GT/s, or by checking that the link is operational at 2.5 GT/s upon re-entering L0. To check that the link is operational, platform will send a configuration read to register 0 (vendor / device ID), and verify that valid data is returned by the DUT. Otherwise, log as DUT failure and note LTSSM state at which link training is stuck.

5. Platform brings the link back to Detect state by initiating fundamental reset to the DUT. Steps 1-4 are repeated, with the platform advertising 5.0 GT/s support. For a 5.0 GT/s capable DUT, at every step, the platform must verify that the link training is successful at 5.0 GT/s.

6. Platform brings the link back to Detect state by initiating fundamental reset to the DUT. Steps 1-4 are repeated, with the platform advertising 8.0 GT/s support. For a 8.0 GT/s capable DUT, at every step, the platform must verify that the link training is successful at 8.0 GT/s.

3.5.2 Root Port Test

The algorithm is the same as an Endpoint except the DUT is a downstream port on a system and the PTC is in Endpoint emulation mode. In Endpoint emulation mode, the fundamental reset is replaced by a power cycle of the system. In Endpoint emulation mode, the platform can only verify that the link is in L0 state at the correct link speed by checking that Idle Symbols are transmitted by the DUT at the correct link speed.

3.5.3 Switch and Bridge Test

For the upstream port of a Switch or Bridge, the Endpoint test is run.
For the downstream port of a Switch or Bridge, the Root Port Test is run.

3.6 Test 56-10 De-emphasis Request During Speed Change

Test Introduction

Tests that a downstream component responds correctly to the request to use -3.5 dB or -6 dB of de-emphasis at 5.0 GT/s during the Recovery state while going through a speed change after the link reaches L0.

Notes:

1. This test applies to Endpoints and Switch Upstream Ports.
2. The test requires the use of the PTC or other piece of test equipment capable of establishing a Link (L0) with the DUT and a real time oscilloscope for measurement.

Procedure:

1. The Test Equipment initiates link training into the L0 state, at 2.5 GT/s data rate.
2. The Test Equipment initiates a speed change to 5.0 GT/s data rate with the Selectable De-emphasis training sequence bit set to request -3.5 dB (bit 6, symbol 4 in TS2).
3. Optionally, read back the Current De-emphasis Level in the Link Status 2 register in all Functions in the Upstream Port of the DUT. This step passes if all values are 1. If any value is 0, report a warning.
4. Capture 1 million unit intervals of data rate at 5.0 GT/s.
5. Measure the mean histogram value of the eye amplitude at the center of the eye for the transition eye ($V_{\text{-3.5dB transition}}$) and non-transition eye ($V_{\text{-3.5dB non-transition}}$).
6. The Test Equipment then brings the link back to the Detect state by initializing fundamental reset to the DUT.
7. The Test Equipment initiates link training into the L0 state, at 2.5 GT/s data rate.
8. The Test Equipment initiates a speed change to 5.0 GT/s data rate with the Selectable De-emphasis training sequence bit set to request -6 dB (bit 6, symbol 4 in TS2).
9. Optionally, read back the Current De-emphasis Level in the Link Status 2 register in all Functions in the Upstream Port of the DUT. This step passes if all values are 0. If any value is 1, report a warning.
10. Capture 1 million unit intervals of data at 5.0 GT/s.

11. Measure the mean histogram value of the eye amplitude at the center of the eye for the transition eye ($V_{\text{-6.0dB transition}}$) and non-transition eye ($V_{\text{-6.0dB non-transition}}$).

12. The test passes if Measured_Change ≥ 0.4 dB. Measured_Change is computed as: Measured_Change = $20 \times \log (V_{\text{-6.0dB transition}} / V_{\text{-6.0dB non-transition}} \times V_{\text{-3.5dB non-transition}} / V_{\text{-3.5dB transition}})$. Note: The base specification requires this be ≥ 0.5 dB. The 0.4 dB criteria allows for measurement error.

3.7 Link Equalization

These tests verify that the DUT will correctly respond to Link Equalization requests to adjust TX EQ presets and coefficients for any legal requests following legal timings. A variety of cases in the TX coefficient space are covered. The test verifies that protocol is correctly followed for each case in the coefficient space that is tested. This test does not verify the correctness of the electrical equalization characteristics of the DUT.

3.7.1 Test 57-10 for Adjusting Initial Preset

3.7.1.1 DUT is a Motherboard or a Downstream Switch Port

1. Perform the steps given in Section A.2.2 to go to the Recovery.RcvrLock state. At this step, the actual link data rate is not 8.0 GT/s.

2. The PTC transmits TS1s with non-PAD link number and lane numbers set during the Configuration states. The speed_change bit is set to 1. All data rates are advertised. If the PTC receives EQ TS1 with any initial preset request with the transmitter preset value in the range of 0xB to 0xF or the receiver preset hint value of 0x7 the DUT will fail the test. After receiving 8 consecutive TS1s or 8 consecutive TS2s with identical link number and lane numbers to the ones being transmitted the PTC transitions to the Recovery.RcvrCfg state.

3. The PTC transmits TS2s with non-PAD link number and non-PAD lane numbers. All data rates are advertised. If the PTC receives EQ TS2 with any initial preset request with the transmitter preset value in the range of 0xB to 0xF or the receiver preset hint value of 0x7 the DUT will fail the test. After receiving 8 consecutive TS2s with speed_change bit set to 1 and 8.0 GT/s data rate advertised, and after transmitting 32 TS2s with speed_change bit set to 1 after receiving the first TS2s and without interruption by an EIEOS, the PTC transitions to the Recovery.Speed state. If the transmission of 32 TS2s is interrupted by an EIEOS, the counting shall be reset to 0 and 32 TS2s shall be retransmitted.

4. In the Recovery.Speed state the PTC shall time out and ends the test.

5. If at any time in the above steps, the PTC link losses contact with the DUT, that Test Case is skipped. (This may be the result of a legal preset that is electrically incompatible with the channel.) However, at least one of the Test Cases must not lose contact in order for the DUT to pass the test.

6. If all the conditions above are met then the DUT passes the test.
7. If any of the conditions above are not met (excluding losing contact with the DUT), log it as DUT's failure.

3.7.1.2 DUT is an Add-in Card or an Upstream Port of a Switch

1. Perform the steps given in Section A.2.2 to reach Recovery.RcvrLock.

2. The PTC transmits TS1s with link and lane numbers set during the Configuration states. The speed_change bit is set to 1. All data rates are advertised. The PTC shall transmit EQ TS1 (symbol 6, bit 7 set to 1) with transmitter preset values (symbol 6, bits 6:3) and receiver preset hint values (symbol 6, bits 2:0) from a Test Case (starting at Test Case 1). All data rates are advertised. After receiving 8 consecutive TS1 or 8 consecutive TS2 with identical link and lane numbers to the one being transmitted the PTC transitions to Recovery.RcvrCfg.

   TEST CASE 1: Transmitter Preset = 0x0, Receiver Preset Hint = 0x0

   TEST CASE 2: Transmitter Preset = 0x1, Receiver Preset Hint = 0x1

   TEST CASE 3: Transmitter Preset = 0x2, Receiver Preset Hint = 0x2

   TEST CASE 4: Transmitter Preset = 0x3, Receiver Preset Hint = 0x3

   TEST CASE 5: Transmitter Preset = 0x4, Receiver Preset Hint = 0x4

   TEST CASE 6: Transmitter Preset = 0x5, Receiver Preset Hint = 0x5

   TEST CASE 7: Transmitter Preset = 0x6, Receiver Preset Hint = 0x6

   TEST CASE 8: Transmitter Preset = 0x7, Receiver Preset Hint = 0x0

   TEST CASE 9: Transmitter Preset = 0x8, Receiver Preset Hint = 0x1

   TEST CASE 10: Transmitter Preset = 0x9, Receiver Preset Hint = 0x2

   TEST CASE 11: Transmitter Preset = 0xA, Receiver Preset Hint = 0x3

3. The PTC transmits TS2s with non-PAD link and lane numbers. All data rates are advertised. The PTC shall transmit EQ TS2 (symbol 6, bit 7 set to 1), with transmitter preset values (symbol 6, bits 6:3) and receiver preset hint values (symbol 6, bits 2:0) from a Test Case (starting at Test Case 1). All data rates are advertised. After receiving 8 consecutive TS2s with speed_change bit set to 1 and 8.0 GT/s advertised, and after transmitting 32 TS2s with speed_change bit set to 1 after receiving the first TS2 and without interruption by an EIEOS, the PTC transitions to Recovery.Speed. If the 32 TS2s are interrupted by an EIEOS, the count is reset and the 32 TS2s are retransmitted. If the received TS2s do not advertise 8.0 GT/s, then the Test Case is aborted, and the test result is reported as skipped.

4. In Recovery.Speed the PTC transmits 1 EIOS and goes to electrical idle. The PTC changes to the highest common advertised data rate within 0.5 ms. The next state is Recovery.RcvrLock at the new data rate.

5. The PTC enters Phase 1 of the Recovery.Equalization state whereas DUT enters Phase 0. The PTC transmits TS1s with EC = 01b, Tx equalization sets the presets it received in the EQ TS2 and reflects its current coefficient values. The PTC should see the DUT reflecting the transmitter preset value it has requested in the TS1s the DUT sends. If the correct requested transmitter preset value is reflected in the received TS1 (symbol 6, bits 6:3), the PTC shall time

out and end the test. The PTC shall repeat steps 1 through 5 again with the next Test Case, until all Test Cases are completed.

6. If at any time in the above steps, the PTC link losses contact with the DUT, that Test Case is skipped. (This may be the result of a legal preset that is electrically incompatible with the channel.) However, at least one of the Test Cases must not lose contact in order for the DUT to pass the test.

7. If all the conditions above are met then the DUT passes the test.

8. If any of the conditions above are not met (excluding losing contact with the DUT), log it as DUT's failure.

3.7.2 Test 58-10 for Adjusting Presets

3.7.2.1 DUT is a Motherboard or a Downstream Switch Port

1. Perform the steps given in Section A.2.2 to reach Recovery.RcvrLock.

2. The PTC transmits TS1s with non-PAD link and lane numbers set during the Configuration states. The speed_change bit is set to 1. All data rates are advertised. On receiving 8 consecutive TS1 or 8 consecutive TS2 with identical link and lane numbers to the one being transmitted the PTC transitions to Recovery.RcvrCfg.

3. The PTC transmits TS2s with non-PAD link and lane numbers. All data rates are advertised. After receiving 8 consecutive TS2s with speed_change bit set to 1 and 8.0 GT/s advertised, and after transmitting 32 TS2s with speed_change bit set to 1 after receiving the first TS2s and without interruption by an EIEOS, the PTC transitions to Recovery.Speed. If the transmission of 32 TS2s is interrupted by an EIEOS, the counting shall be reset to 0 and 32 TS2s shall be retransmitted.

4. In Recovery.Speed the PTC transmits 1 EIOS and goes to electrical idle. The PTC shall change to 8.0 GT/s data rate within 0.5 ms. The next state is Recovery.RcvrLock.

5. The DUT enters Phase 1 of the Recovery.Equalization state whereas the PTC enters Phase 0. The PTC transmits TS1s with EC = 00b, with Transmitter Preset containing the value it received in the EQ TS2s, and with Pre-cursor, Cursor, and Post-cursor coefficient values associated with that Transmitter Preset. PTC should record per lane the FS (symbol 7, bits 5-0) and LF (symbol 8, bits 5-0) values received in two consecutive TS1s with EC = 01b (for use in step 7). If the recorded LF value is greater than the FS value on any lane, the DUT fails. After receiving 2 consecutive TS1s with EC = 01b within 2 ms the PTC transitions to Phase 1. Alternately, if the PTC receives 8 consecutive TS1s with EC = 00b, the PTC goes to the Recovery.RcvrLock state and ends the test (this is not a failure, but instead the test result is reported as skipped). If the PTC does not receive either 2 consecutive TS1s with EC = 01b or 8 consecutive TS1s with EC = 00b, within 12 ms, the DUT fails.

6. In Phase 1, the PTC transmits TS1s with EC=01b, with preset set to the one requested in the EQ TS2s it received from the DUT and reflects its current FS, LF, and Post-cursor Coefficient values. After receiving 2 consecutive TS1s with EC = 10b within 2 ms the PTC transitions to

Phase 2. Alternately, if the PTC receives 8 consecutive TS1s with EC = 00b, the PTC goes to the Recovery.RcvrLock state and ends the test (this is not a failure, but instead the test result is reported as skipped). If the PTC does not receive either 2 consecutive TS1s with EC = 10b or 8 consecutive TS1s with EC = 00b, within 12 ms, the DUT fails.

7. In Phase 2 the PTC transmits TS1 with EC = 10b, with the Use Preset bit (symbol 6, bit 7) set to a Test Case value (starting at Test Case 1), and transmitter preset values (symbol 6, bits 6:3) from a Test Case (starting at Test Case 1). The PTC must transmit these TS1 for at least 1 µs. The PTC shall obtain new block alignment every time it receives the new coefficients. The PTC should wait at least 1 µs and then check that the received TS1 contains the expected values for the Test Case (starting at Test Case 1). When the Expected Pre-Cursor, Cursor and Post-Cursor values are "any valid", the received TS1 matches if all of the following are true (using FS and LF values captured in Step 5):

    a) Pre-Cursor <= Floor (FS/4)
    b) Pre-Cursor +Cursor +Post-Cursor = FS
    c) Cursor – Pre-Cursor – Post-Cursor >= LF

If this check fails and the requested Transmitter Preset is not Reserved (e.g. 0x0 to 0xA), the DUT fails the test. If this check fails and the requested Transmitter Preset is Reserved (e.g. 0xB to 0xF), the DUT is issued a warning.

TEST CASE 1: Transmitter Preset = 0x0, Use Preset = 1, Pre-Cursor = 0x01, Cursor = FS, Post-Cursor = 0x01

Expected Transmitter Preset = 0x0

Expected Reject Coefficient Values = 0

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 2: Transmitter Preset = 0x1, Use Preset = 1, Pre-Cursor = Floor(FS/4)+1, Cursor = FS – Floor(FS/4) – 1, Post-Cursor = 0x0

Expected Transmitter Preset = 0x1

Expected Reject Coefficient Values = 0

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 3: Transmitter Preset = 0x2, Use Preset = 1, Pre-Cursor = Floor(FS/4), Cursor = FS – 2*Floor(FS/4), Post-Cursor = Floor(FS/4)

Expected Transmitter Preset = 0x2

Expected Reject Coefficient Values = 0

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 4: Transmitter Preset = 0x3, Use Preset = 1, Pre-Cursor = 0x01, Cursor = FS, Post-Cursor = 0x01

Expected Transmitter Preset = 0x3

Expected Reject Coefficient Values = 0

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 5: Transmitter Preset = 0x4, Use Preset = 1, Pre-Cursor = Floor(FS/4)+1, Cursor = FS – Floor(FS/4) – 1, Post-Cursor = 0x0

Expected Transmitter Preset = 0x4

Expected Reject Coefficient Values = 0

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 6: Transmitter Preset = 0x5, Use Preset = 1, Pre-Cursor = Floor(FS/4), Cursor = FS – 2*Floor(FS/4), Post-Cursor = Floor(FS/4)

Expected Transmitter Preset = 0x5

Expected Reject Coefficient Values = 0

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 7: Transmitter Preset = 0x6, Use Preset = 1, Pre-Cursor = 0x01, Cursor = FS, Post-Cursor = 0x01

Expected Transmitter Preset = 0x6

Expected Reject Coefficient Values = 0

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 8: Transmitter Preset = 0x7, Use Preset = 1, Pre-Cursor = Floor(FS/4)+1, Cursor = FS – Floor(FS/4) – 1, Post-Cursor = 0x0

Expected Transmitter Preset = 0x7

Expected Reject Coefficient Values = 0

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 9: Transmitter Preset = 0x8, Use Preset = 1, Pre-Cursor = Floor(FS/4), Cursor = FS – 2*Floor(FS/4), Post-Cursor = Floor(FS/4)

Expected Transmitter Preset = 0x8

Expected Reject Coefficient Values = 0

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 10: Transmitter Preset = 0x9, Use Preset = 1, Pre-Cursor = 0x01, Cursor = FS, Post-Cursor = 0x01

Expected Transmitter Preset = 0x9

Expected Reject Coefficient Values = 0

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 11: Transmitter Preset = 0xA, Use Preset = 1, Pre-Cursor = Floor(FS/4)+1, Cursor = FS – Floor(FS/4) – 1, Post-Cursor = 0x0

Expected Transmitter Preset = 0xA

Expected Reject Coefficient Values = 0

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 12: Transmitter Preset = 0xB, Use Preset = 1, Pre-Cursor = Floor(FS/4), Cursor = FS – 2*Floor(FS/4), Post-Cursor = Floor(FS/4)

Expected Transmitter Preset = 0xB

Expected Reject Coefficient Values = 1

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 13: Transmitter Preset = 0xC, Use Preset = 1, Pre-Cursor = 0x01, Cursor = FS, Post-Cursor = 0x01

Expected Transmitter Preset = 0xC

Expected Reject Coefficient Values = 1

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 14: Transmitter Preset = 0xD, Use Preset = 1, Pre-Cursor = Floor(FS/4)+1, Cursor = FS – Floor(FS/4) – 1, Post-Cursor = 0x0

Expected Transmitter Preset = 0xD

Expected Reject Coefficient Values = 1

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 15: Transmitter Preset = 0xE, Use Preset = 1, Pre-Cursor = Floor(FS/4), Cursor = FS – 2*Floor(FS/4), Post-Cursor = Floor(FS/4)

Expected Transmitter Preset = 0xE

Expected Reject Coefficient Values = 1

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 16: Transmitter Preset = 0xF, Use Preset = 1, Pre-Cursor = 0x01, Cursor = FS, Post-Cursor = 0x01

Expected Transmitter Preset = 0xF

Expected Reject Coefficient Values = 1

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

Note: An earlier draft of this specification defined Test Case 17. This has since been removed. There is no harm in running this test case. If run, it does not affect the pass/fail status of the DUT.

8. In Phase 3 the PTC transmit TS1s with EC=11b, reflecting its current coefficient values, signaling the DUT to transition to Phase 3. If the PTC receives 2 consecutive TS1s with EC = 11b and requesting a valid new preset or coefficient, it processes these normally. If the PTC receives any preset request in the range of 0xB to 0xF the PTC will reject this (Reject Coefficient is set to 1 in the PTC's response) and the DUT will fail the test. If the PTC receives any coefficient request that is illegal for its advertised FS, LF range the PTC will reject this (Reject Coefficient is set to 1 in the PTC's response) and the DUT will not fail the test and testing will continue. Otherwise when the PTC receives 2 consecutive TS1 with EC = 00b, the PTC transitions to the Recovery.RcvrLock state, then the PTC goes to electrical idle and times out the DUT and ends the test. If the PTC does not receive either 2 consecutive TS1s with EC = 11b or 2 consecutive TS1s with EC = 00b, within 32 ms, the DUT fails.
9. Steps 1 through 8 are repeated until all Test Cases are tested.
10. If at any time in the above steps, when the requested Transmitter Preset is not Reserved (e.g. 0x0 to 0xA) and the PTC link loses contact with the DUT, that Test Case is skipped. (This may be the result of a legal preset that is electrically incompatible with the channel.) However, at least one of these Test Cases must not lose contact in order for the DUT to pass the test.

    If at any time in the above steps, when the requested Transmitter Preset is Reserved (e.g. 0xB to 0xF) and the PTC link loses contact with the DUT, the DUT fails if the previous Test Case did not lose contact with the DUT and the Test Case is skipped if the previous Test Case also lost contact with the DUT.
11. If all the conditions above are met, then the DUT passes the test.
12. If any of the conditions above are not met (excluding losing contact with the DUT pursuant to step 10, or warnings), log it as DUT's failure.

3.7.2.2 DUT is an Add-in Card or an Upstream Port of a Switch

1. Perform steps given in Section A.2.2 to reach Recovery.RcvrLock.
2. The PTC transmits TS1s with link and lane numbers set during the Configuration states. The speed_change bit is set to 1. All data rates are advertised. The PTC shall transmit EQ TS1 (symbol 6, bit 7 set to 1) with transmitter preset values (symbol 6, bits 6:3) of 0x0 and receiver preset hint values (symbol 6, bits 2:0) of 0x0. All data rates are advertised. After receiving 8 consecutive TS1 or 8 consecutive TS2 with identical link and lane numbers to the one being transmitted the PTC transitions to Recovery.RcvrCfg.
3. The PTC transmits TS2s with non-PAD link and lane numbers. All data rates are advertised. The PTC shall transmit EQ TS2 (symbol 6, bit 7 set to 1) with transmitter preset values (symbol 6, bits 6:3) of 0x0 and receiver preset hint values (symbol 6, bits 2:0) of 0x0. All data rates are advertised. After receiving 8 consecutive TS2s with speed_change bit set to 1 and 8.0 GT/s advertised, and after transmitting 32 TS2s with speed_change bit set to 1 and all data rates advertised after receiving the first TS2 and without interruption by an EIEOS, the PTC transitions to Recovery.Speed. If the 32 TS2s are interrupted by an EIEOS, the count is reset

and the 32 TS2s are retransmitted. If the received TS2s do not advertise 8.0 GT/s, then the test case is aborted, and the test result is reported as skipped.

4. In Recovery.Speed the PTC transmits 1 EIOS and goes to electrical idle. The PTC changes to the highest common advertised data rates within 0.5 ms. The next state is Recovery.RcvrLock at the new data rate.

5. The PTC enters Phase 1 of the Recovery.Equalization state whereas the DUT enters Phase 0. The PTC transmits TS1s with EC = 01b, with Transmitter Preset = 0x0, and its current FS, LF, and Post-cursor Coefficient values. The PTC should first see the DUT sending TS1s with EC = 00b reflecting the Transmitter Preset value it requested in the EQ TS2s. The PTC should record per lane the FS (symbol 7, bits 5-0) and LF (symbol 8, bits 5-0) values received in two consecutive TS1s with EC = 01b (for use in step 7). If the recorded LF value is greater than the FS value on any lane, the DUT fails. After receiving 2 consecutive TS1s with EC = 01b within 2 ms the PTC should transition to Phase 2. If the PTC does not receive 2 consecutive TS1s with EC = 01b within 24 ms, the DUT fails.

6. In Phase 2, the PTC transmits TS1s with EC = 10b, reflecting its current coefficient values. If the PTC receives 2 consecutive TS1s with EC = 10b and requesting a valid new preset or coefficient, it processes these normally. If the PTC receives any preset request in the range of 0xB to 0xF the PTC will reject this (Reject Coefficient is set to 1 in the PTC's response) and the DUT will fail the test. If the PTC receives any coefficient request that is illegal for its advertised FS, LF range the PTC will reject this (Reject Coefficient is set to 1 in the PTC's response) however the DUT will not fail the test and testing will continue. Otherwise, when the PTC receives 2 consecutive TS2 with EC = 11b, the PTC transitions to Phase 3. If the PTC does not receive 2 consecutive TS1s with EC = 11b within 32 ms, the DUT fails.

7. In Phase 3, the PTC transmits TS1s with EC=11b, with the Use Preset bit (symbol 6, bit 7) and Transmitter Preset values (symbol 6, bits 6:3) from a Test Case (starting at Test Case 1). The PTC must transmit these TS1 for at least 1 µs. The PTC shall obtain new block alignment every time it receives the new coefficients. The PTC should wait at least 1 µs and then check that the received TS1 contains the expected values for the Test Case (starting at Test Case 1). When the Expected Pre-Cursor, Cursor and Post-Cursor values are "any valid", the received TS1 matches if all of the following are true (using FS and LF values captured in Step 5):

    a) Pre-Cursor <= Floor (FS/4)
    b) Pre-Cursor +Cursor +Post-Cursor = FS
    c) Cursor – Pre-Cursor – Post-Cursor >= LF

    If this check fails and the requested preset is not Reserved (e.g. 0x0 to 0xA), the DUT fails the test. If this check fails and the requested preset is Reserved (e.g. 0x0B to 0xF), the DUT is issued a warning. The range of valid values is given by the following:

    TEST CASE 1: Transmitter Preset = 0x0, Use Preset = 1, Pre-Cursor = 0x01, Cursor = FS, Post-Cursor = 0x01

    Expected Transmitter Preset = 0x0

    Expected Reject Coefficient Values = 0

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 2: Transmitter Preset = 0x1, Use Preset = 1, Pre-Cursor = Floor(FS/4)+1, Cursor = FS – Floor(FS/4) – 1, Post-Cursor = 0x0

Expected Transmitter Preset = 0x1

Expected Reject Coefficient Values = 0

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 3: Transmitter Preset = 0x2, Use Preset = 1, Pre-Cursor = Floor(FS/4), Cursor = FS – 2*Floor(FS/4), Post-Cursor = Floor(FS/4)

Expected Transmitter Preset = 0x2

Expected Reject Coefficient Values = 0

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 4: Transmitter Preset = 0x3, Use Preset = 1, Pre-Cursor = 0x01, Cursor = FS, Post-Cursor = 0x01

Expected Transmitter Preset = 0x3

Expected Reject Coefficient Values = 0

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 5: Transmitter Preset = 0x4, Use Preset = 1, Pre-Cursor = Floor(FS/4)+1, Cursor = FS – Floor(FS/4) – 1, Post-Cursor = 0x0

Expected Transmitter Preset = 0x4

Expected Reject Coefficient Values = 0

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 6: Transmitter Preset = 0x5, Use Preset = 1, Pre-Cursor = Floor(FS/4), Cursor = FS – 2*Floor(FS/4), Post-Cursor = Floor(FS/4)

Expected Transmitter Preset = 0x5

Expected Reject Coefficient Values = 0

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 7: Transmitter Preset = 0x6, Use Preset = 1, Pre-Cursor = 0x01, Cursor = FS, Post-Cursor = 0x01

Expected Transmitter Preset = 0x6

Expected Reject Coefficient Values = 0

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 8: Transmitter Preset = 0x7, Use Preset = 1, Pre-Cursor = Floor(FS/4)+1, Cursor = FS – Floor(FS/4) – 1, Post-Cursor = 0x0

Expected Transmitter Preset = 0x7

Expected Reject Coefficient Values = 0

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 9: Transmitter Preset = 0x8, Use Preset = 1, Pre-Cursor = Floor(FS/4), Cursor = FS – 2*Floor(FS/4), Post-Cursor = Floor(FS/4)

Expected Transmitter Preset = 0x8

Expected Reject Coefficient Values = 0

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 10: Transmitter Preset = 0x9, Use Preset = 1, Pre-Cursor = 0x01, Cursor = FS, Post-Cursor = 0x01

Expected Transmitter Preset = 0x9

Expected Reject Coefficient Values = 0

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 11: Transmitter Preset = 0xA, Use Preset = 1, Pre-Cursor = Floor(FS/4)+1, Cursor = FS – Floor(FS/4) – 1, Post-Cursor = 0x0

Expected Transmitter Preset = 0xA

Expected Reject Coefficient Values = 0

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 12: Transmitter Preset = 0xB, Use Preset = 1, Pre-Cursor = Floor(FS/4), Cursor = FS – Floor(FS/4), Post-Cursor = Floor(FS/4)

Expected Transmitter Preset = 0xB

Expected Reject Coefficient Values = 1

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 13: Transmitter Preset = 0xC, Use Preset = 1, Pre-Cursor = 0x01, Cursor = FS, Post-Cursor = 0x01

Expected Transmitter Preset = 0xC

Expected Reject Coefficient Values = 1

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 14: Transmitter Preset = 0xD, Use Preset = 1, Pre-Cursor = Floor(FS/4)+1, Cursor = FS – Floor(FS/4) – 1, Post-Cursor = 0x0

Expected Transmitter Preset = 0xD

Expected Reject Coefficient Values = 1

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 15: Transmitter Preset = 0xE, Use Preset = 1, Pre-Cursor = Floor(FS/4), Cursor = FS – Floor(FS/4), Post-Cursor = Floor(FS/4)

Expected Transmitter Preset = 0xE

Expected Reject Coefficient Values = 1

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

TEST CASE 16: Transmitter Preset = 0xF, Use Preset = 1, Pre-Cursor = 0x01, Cursor = FS, Post-Cursor = 0x01

Expected Transmitter Preset = 0xF

Expected Reject Coefficient Values = 1

Expected Pre-Cursor, Cursor, and Post-Cursor = any valid

Note: An earlier draft of this specification defined Test Case 17. This has since been removed. There is no harm in running this test case. If run, it does not affect the pass/fail status of the DUT.

8. Steps 1 through 7 are repeated until all Test Cases are tested, and then the PTC transitions to the Recovery.RcvrLock state, then the PTC goes to electrical idle and times out the DUT and ends the test. Note: The test must complete all Test Cases within 24 ms; otherwise the DUT may time out and exit link equalization.

9. If at any time in the above steps, when the requested Transmitter Preset is not Reserved (e.g. 0x0 to 0xA) and the PTC link loses contact with the DUT, that Test Case is skipped. (This may be the result of a legal preset that is electrically incompatible with the channel.) However, at least one of these Test Cases must not lose contact in order for the DUT to pass the test.

   If at any time in the above steps, when the requested Transmitter Preset is Reserved (e.g. 0xB to 0xF) and the PTC link loses contact with the DUT, the DUT fails if the previous Test Case did not lose contact with the DUT and the Test Case is skipped if the previous Test Case also lost contact with the DUT.

10. If all the conditions above are met, then the DUT passes the test.

11. If any of the conditions above are not met (excluding losing contact with the DUT pursuant to step 9, or warnings), log it as DUT's failure.

3.7.3 Test 59-10 for Adjusting Coefficients

3.7.3.1 DUT is a Motherboard or a Downstream Switch Port

1. Perform the steps given in Section A.2.2 to reach Recovery.RcvrLock.

2. The PTC transmits TS1s with non-PAD link and lane numbers set during the Configuration states. The speed_change bit is set to 1. All data rates are advertised. After receiving 8 consecutive TS1 or 8 consecutive TS2 with identical link and lane numbers to the one being transmitted the PTC transitions to Recovery.RcvrCfg.

3. The PTC transmits TS2s with non-PAD link and lane numbers. All data rates are advertised. After receiving 8 consecutive TS2s with speed_change bit set to 1 and 8.0 GT/s advertised, and after transmitting 32 TS2s with speed_change bit set to 1 and all data rates advertised after receiving the first TS2s and without interruption by an EIEOS, the PTC transitions to Recovery.Speed. If the transmission of 32 TS2s is interrupted by an EIEOS, the counting shall be reset to 0 and 32 TS2s shall be retransmitted. If the received TS2s do not advertise 8.0 GT/s, then the test case is aborted, and the test result is reported as skipped.

4. In Recovery.Speed the PTC transmits 1 EIOS and goes to electrical idle. The PTC shall change to 8.0 GT/s data rate within 0.5 ms. The next state is Recovery.RcvrLock.

5. The DUT enters Phase 1 of the Recovery.Equalization state whereas the PTC enters Phase 0. The PTC transmits TS1s with EC = 00b, Tx equalization sets the presets it received in the EQ TS2s, and reflects its current coefficient values. The PTC should record per lane the FS (symbol 7, bits 5-0) and LF (symbol 8, bits 5-0) values received in two consecutive TS1s with EC = 01b (for use in step 7). If the recorded LF value is greater than the FS value on any lane, the DUT fails. After receiving 2 consecutive TS1s with EC = 01b within 2 ms the PTC transitions to Phase 1. Alternately, if the PTC receives 8 consecutive TS1s with EC = 00b, the PTC goes to the Recovery.RcvrLock state and ends the test (this is not a failure, but instead the test result is reported as skipped). If the PTC does not receive either 2 consecutive TS1s with EC = 01b or 8 consecutive TS1s with EC = 00b, within 12 ms, the DUT fails.

6. In Phase 1, the PTC transmits TS1s with EC=01b, with preset set to the one requested in the EQ TS2s it received from the DUT and reflects its current FS, LF, and post-coefficient values. After receiving 2 consecutive TS1s with EC = 10b within 2 ms the PTC transitions to Phase 2. Alternately, if the PTC receives 8 consecutive TS1s with EC = 00b, the PTC goes to the Recovery.RcvrLock state and ends the test (this is not a failure, but instead the test result is reported as skipped). If the PTC does not receive either 2 consecutive TS1s with EC = 10b or 8 consecutive TS1s with EC = 00b, within 12 ms, the DUT fails.

7. In Phase 2 the PTC transmits TS1 with EC = 10b, with the Use Preset bit (symbol 6, bit 7) set to a Test Case value (starting at Test Case 1), and transmitter pre-cursor/cursor/post-cursor coefficient values (symbol 7, bits 5:0/symbol 8, bits 5:0/symbol 9, bits 5:0) from a Test Case (starting at Test Case 1). The PTC must transmit these TS1 for at least 1 µs. The PTC shall obtain new block alignment every time it receives the new coefficients. The PTC should wait at least 1 µs and then check that the received TS1 contains the expected Reject Coefficient, Pre-cursor, Cursor, and Post-cursor coefficient values (symbol 9 bit 6, symbol 7 bits 5:0,symbol 8 bits 5:0, and symbol 9 bits 5:0 respectively) for the Test Case (starting at Test Case 1). If this

check fails and the Test Case's Expected Reject Coefficient Values is 0, the DUT fails the test. If this check fails and the Test Case's Expected Reject Coefficient Values is 1, the DUT is issued a warning. The range of valid values is given by the following (where [FS] and [LF] are the values captured in step 5):

TEST CASE 1: Pre-cursor Coefficient (C_{-1}) = 0x00, Cursor Coefficient (C_0) = 0x00, Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

Expected Pre-cursor Coefficient = 0x00

Expected Cursor Coefficient = 0x00

Expected Post-cursor Coefficient = 0x00

Expected Reject Coefficient Values = 1 (the coefficients add up to less than 12)

TEST CASE 2: Pre-cursor Coefficient (C_{-1}) = 0x00, Cursor Coefficient (C_0) = 0x0B, Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

Expected Pre-cursor Coefficient = 0x00

Expected Cursor Coefficient = 0x0B

Expected Post-cursor Coefficient = 0x00

Expected Reject Coefficient Values = 1 (the coefficients add up to less than 12)

TEST CASE 3: Pre-cursor Coefficient (C_{-1}) = 0x0B, Cursor Coefficient (C_0) = 0x00, Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0B

Expected Cursor Coefficient = 0x00

Expected Post-cursor Coefficient = 0x00

Expected Reject Coefficient Values = 1 (the coefficients add up to less than 12)

TEST CASE 4: Pre-cursor Coefficient (C_{-1}) = 0x00, Cursor Coefficient (C_0) = 0x00, Post-cursor Coefficient (C_{+1}) = 0x0B, Use Preset = 0

Expected Pre-cursor Coefficient = 0x00

Expected Cursor Coefficient = 0x00

Expected Post-cursor Coefficient = 0x0B

Expected Reject Coefficient Values = 1 (the coefficients add up to less than 12)

TEST CASE 5: Pre-cursor Coefficient (C_{-1}) = 0x01, Cursor Coefficient (C_0) = 0x09, Post-cursor Coefficient (C_{+1}) = 0x01, Use Preset = 0

Expected Pre-cursor Coefficient = 0x01

Expected Cursor Coefficient = 0x09

Expected Post-cursor Coefficient = 0x01

Expected Reject Coefficient Values = 1 (the coefficients add up to less than 12)

TEST CASE 6: Pre-cursor Coefficient (C_{-1}) = 0x00, Cursor Coefficient (C_0) = [FS], Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

Expected Pre-cursor Coefficient = 0x00

Expected Cursor Coefficient = [FS]

Expected Post-cursor Coefficient = 0x00

Expected Reject Coefficient Values = 0

TEST CASE 7 (only if [FS] < 63): Pre-cursor Coefficient (C_{-1}) = 0x00, Cursor Coefficient (C_0) = [FS]+1, Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

Expected Pre-cursor Coefficient = 0x00

Expected Cursor Coefficient = [FS]+1

Expected Post-cursor Coefficient = 0x00

Expected Reject Coefficient Values = 1 (the coefficients add up to more than [FS])

TEST CASE 8: Pre-cursor Coefficient (C_{-1}) = 0x00, Cursor Coefficient (C_0) = [FS]-1, Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

Expected Pre-cursor Coefficient = 0x00

Expected Cursor Coefficient = [FS]-1

Expected Post-cursor Coefficient = 0x00

Expected Reject Coefficient Values = 1 (the coefficients add up to less than [FS])

TEST CASE 9: Pre-cursor Coefficient (C_{-1}) = 0x01, Cursor Coefficient (C_0) = [FS], Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

Expected Pre-cursor Coefficient = 0x01

Expected Cursor Coefficient = [FS]

Expected Post-cursor Coefficient = 0x00

Expected Reject Coefficient Values = 1 (the coefficients add up to more than [FS])

TEST CASE 10: Pre-cursor Coefficient (C_{-1}) = 0x00, Cursor Coefficient (C_0) = [FS], Post-cursor Coefficient (C_{+1}) = 0x01, Use Preset = 0

Expected Pre-cursor Coefficient = 0x00

Expected Cursor Coefficient = [FS]

Expected Post-cursor Coefficient = 0x01

Expected Reject Coefficient Values = 1 (the coefficients add up to more than [FS])

TEST CASE 11: Pre-cursor Coefficient (C_{-1}) = 0x01, Cursor Coefficient (C_0) = [FS]-1, Post-cursor Coefficient (C_{+1}) = 0x01, Use Preset = 0

Expected Pre-cursor Coefficient = 0x01

Expected Cursor Coefficient = [FS]-1

Expected Post-cursor Coefficient = 0x01

Expected Reject Coefficient Values = 1 (the coefficients add up to more than [FS])

TEST CASE 12: Pre-cursor Coefficient (C_{-1}) = 0x01, Cursor Coefficient (C_0) = [FS]-2, Post-cursor Coefficient (C_{+1}) = 0x01, Use Preset = 0

Expected Pre-cursor Coefficient = 0x01

Expected Cursor Coefficient = [FS]-2

Expected Post-cursor Coefficient = 0x01

Expected Reject Coefficient Values = 0 if ([FS]-4) >= [LF]; 1 if ([FS]-4) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 13: Pre-cursor Coefficient (C_{-1}) = 0x01, Cursor Coefficient (C_0) = [FS]-3, Post-cursor Coefficient (C_{+1}) = 0x01, Use Preset = 0

Expected Pre-cursor Coefficient = 0x01

Expected Cursor Coefficient = [FS]-3

Expected Post-cursor Coefficient = 0x01

Expected Reject Coefficient Values = 1 (the coefficients add up to less than [FS])

TEST CASE 14: Pre-cursor Coefficient (C_{-1}) = 0x02, Cursor Coefficient (C_0) = [FS]-4, Post-cursor Coefficient (C_{+1}) = 0x02, Use Preset = 0

Expected Pre-cursor Coefficient = 0x02

Expected Cursor Coefficient = [FS]-4

Expected Post-cursor Coefficient = 0x02

Expected Reject Coefficient Values = 0 if ([FS]-8) >= [LF]; 1 if ([FS]-8) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 15: Pre-cursor Coefficient (C_{-1}) = 0x03, Cursor Coefficient (C_0) = [FS]-6, Post-cursor Coefficient (C_{+1}) = 0x03, Use Preset = 0

Expected Pre-cursor Coefficient = 0x03

Expected Cursor Coefficient = [FS]-6

Expected Post-cursor Coefficient = 0x03

Expected Reject Coefficient Values = 0 if ([FS]-12) >= [LF]; 1 if ([FS]-12) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 16: Pre-cursor Coefficient (C_{-1}) = 0x04, Cursor Coefficient (C_0) = [FS]-8, Post-cursor Coefficient (C_{+1}) = 0x04, Use Preset = 0

Expected Pre-cursor Coefficient = 0x04

Expected Cursor Coefficient = [FS]-8

Expected Post-cursor Coefficient = 0x04

Expected Reject Coefficient Values = 0 if ([FS]-16) >= [LF]; 1 if ([FS]-16) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 17: Pre-cursor Coefficient (C_{-1}) = 0x05, Cursor Coefficient (C_0) = [FS]-10, Post-cursor Coefficient (C_{+1}) = 0x05, Use Preset = 0

Expected Pre-cursor Coefficient = 0x05

Expected Cursor Coefficient = [FS]-10

Expected Post-cursor Coefficient = 0x05

Expected Reject Coefficient Values = 0 if ([FS]-20) >= [LF]; 1 if ([FS]-20) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 18: Pre-cursor Coefficient (C_{-1}) = 0x06, Cursor Coefficient (C_0) = [FS]-12, Post-cursor Coefficient (C_{+1}) = 0x06, Use Preset = 0

Expected Pre-cursor Coefficient = 0x06

Expected Cursor Coefficient = [FS]-12

Expected Post-cursor Coefficient = 0x06

Expected Reject Coefficient Values = 0 if ([FS]-24) >= [LF]; 1 if ([FS]-24) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 19 (only if [FS]-14 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x07, Cursor Coefficient (C_0) = [FS]-14, Post-cursor Coefficient (C_{+1}) = 0x07, Use Preset = 0

Expected Pre-cursor Coefficient = 0x07

Expected Cursor Coefficient = [FS]-14

Expected Post-cursor Coefficient = 0x07

Expected Reject Coefficient Values = 0 if ([FS]-28) >= [LF]; 1 if ([FS]-28) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 20 (only if [FS]-16 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x08, Cursor Coefficient (C_0) = [FS]-16, Post-cursor Coefficient (C_{+1}) = 0x08, Use Preset = 0

Expected Pre-cursor Coefficient = 0x08

Expected Cursor Coefficient = [FS]-16

Expected Post-cursor Coefficient = 0x08

Expected Reject Coefficient Values = 0 if ([FS]-32) >= [LF]; 1 if ([FS]-32) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 21h (only if [FS]-18 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x09, Cursor Coefficient (C_0) = [FS]-18, Post-cursor Coefficient (C_{+1}) = 0x09, Use Preset = 0

Expected Pre-cursor Coefficient = 0x09

Expected Cursor Coefficient = [FS]-18

Expected Post-cursor Coefficient = 0x09

Expected Reject Coefficient Values = 0 if ([FS]-36) >= [LF]; 1 if ([FS]-36) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 22 (only if [FS]-20 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0A, Cursor Coefficient (C_0) = [FS]-20, Post-cursor Coefficient (C_{+1}) = 0x0A, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0A

Expected Cursor Coefficient = [FS]-20

Expected Post-cursor Coefficient = 0x0A

Expected Reject Coefficient Values = 0 if ([FS]-40) >= [LF]; 1 if ([FS]-40) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 23 (only if [FS]-22 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0B, Cursor Coefficient (C_0) = [FS]-22, Post-cursor Coefficient (C_{+1}) = 0x0B, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0B

Expected Cursor Coefficient = [FS]-22

Expected Post-cursor Coefficient = 0x0B

Expected Reject Coefficient Values = 0 if ([FS]-44) >= [LF]; 1 if ([FS]-44) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 24 (only if [FS]-24 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0C, Cursor Coefficient (C_0) = [FS]-24, Post-cursor Coefficient (C_{+1}) = 0x0C, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0C

Expected Cursor Coefficient = [FS]-24

Expected Post-cursor Coefficient = 0x0C

Expected Reject Coefficient Values = 0 if ([FS]-48) >= [LF]; 1 if ([FS]-48) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 25 (only if [FS]-26 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0D, Cursor Coefficient (C_0) = [FS]-26, Post-cursor Coefficient (C_{+1}) = 0x0D, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0D

Expected Cursor Coefficient = [FS]-26

Expected Post-cursor Coefficient = 0x0D

Expected Reject Coefficient Values = 0 if ([FS]-52) >= [LF]; 1 if ([FS]-52) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 26 (only if [FS]-28 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0E, Cursor Coefficient (C_0) = [FS]-28, Post-cursor Coefficient (C_{+1}) = 0x0E, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0E

Expected Cursor Coefficient = [FS]-28

Expected Post-cursor Coefficient = 0x0E

Expected Reject Coefficient Values = 0 if ([FS]-56) >= [LF]; 1 if ([FS]-56) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 27 (only if [FS]-30 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0F, Cursor Coefficient (C_0) = [FS]-30, Post-cursor Coefficient (C_{+1}) = 0x0F, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0F

Expected Cursor Coefficient = [FS]-30

Expected Post-cursor Coefficient = 0x0F

Expected Reject Coefficient Values = 0 if ([FS]-60) >= [LF]; 1 if ([FS]-60) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 28 (only if [FS] > [LF]): Pre-cursor Coefficient (C_{-1}) = 0x00, Cursor Coefficient (C_0) = [LF], Post-cursor Coefficient (C_{+1}) = [FS] - [LF], Use Preset = 0

Expected Pre-cursor Coefficient = 0x00

Expected Cursor Coefficient = [LF]

Expected Post-cursor Coefficient = [FS]-[LF]

Expected Reject Coefficient Values = 1 (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 29 (only if [FS] > [LF]): Pre-cursor Coefficient (C_{-1}) = [FS] - [LF], Cursor Coefficient (C_0) = [LF], Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

Expected Pre-cursor Coefficient = [FS]-[LF]

Expected Cursor Coefficient = [LF]

Expected Post-cursor Coefficient = 0x00

Expected Reject Coefficient Values = 1 (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 30 (only if [FS] > [LF]): Pre-cursor Coefficient (C_{-1}) = Floor(([FS] - [LF])/2), Cursor Coefficient (C_0) = [LF], Post-cursor Coefficient (C_{+1}) = Ceiling(([FS] - [LF])/2), Use Preset = 0

Expected Pre-cursor Coefficient = Floor(([FS]-[LF])/2)

Expected Cursor Coefficient = [LF]

Expected Post-cursor Coefficient = Ceiling(([FS]-[LF])/2)

Expected Reject Coefficient Values = 1 (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 31: Pre-cursor Coefficient (C_{-1}) = 0x00, Cursor Coefficient (C_0) = [LF]-1, Post-cursor Coefficient (C_{+1}) = [FS]-[LF]+1, Use Preset = 0

Expected Pre-cursor Coefficient = 0x00

Expected Cursor Coefficient = [LF]-1

Expected Post-cursor Coefficient = [FS]-[LF]+1

Expected Reject Coefficient Values = 1 (the post-cursor coefficient is greater than the cursor coefficient minus [LF])

TEST CASE 32: Pre-cursor Coefficient (C_{-1}) = [FS]-[LF]+1, Cursor Coefficient (C_0) = [LF]-1, Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

Expected Pre-cursor Coefficient = [FS]-[LF]+1

Expected Cursor Coefficient = [LF]-1

Expected Post-cursor Coefficient = 0x00

Expected Reject Coefficient Values = 1 (the pre-cursor coefficient is greater than the cursor coefficient minus [LF])

TEST CASE 33 (only if [FS] > [LF]): Pre-cursor Coefficient (C_{-1}) = Floor(([FS]-[LF])/2)+1, Cursor Coefficient (C_0) = [LF]+1, Post-cursor Coefficient (C_{+1}) = Ceiling(([FS]-[LF])/2)+1, Use Preset = 0

Expected Pre-cursor Coefficient = Floor(([FS]-[LF])/2)+1

Expected Cursor Coefficient = [LF]+1

Expected Post-cursor Coefficient = Ceiling(([FS]-[LF])/2)+1

Expected Reject Coefficient Values = 1 (the coefficients add up to more than [FS])

TEST CASE 34: Pre-cursor Coefficient (C_{-1}) = Floor([FS]/4), Cursor Coefficient (C_0) = [FS] – Floor([FS]/4), Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

Expected Pre-cursor Coefficient = Floor([FS]/4)

Expected Cursor Coefficient = [FS]-Floor([FS]/4)

Expected Post-cursor Coefficient = 0x00

Expected Reject Coefficient Values = 0

TEST CASE 35: Pre-cursor Coefficient (C_{-1}) = (Floor([FS]/4))+1, Cursor Coefficient (C_0) = [FS] – ((Floor([FS]/4))+1), Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

Expected Pre-cursor Coefficient = (Floor([FS]/4))+1

Expected Cursor Coefficient = [FS]-((Floor([FS]/4))+1)

Expected Post-cursor Coefficient = 0x00

Expected Reject Coefficient Values = 1 (the pre-cursor coefficient is greater than [FS]/4)

TEST CASE 36: Pre-cursor Coefficient (C_{-1}) = (Floor([FS]/4))-1, Cursor Coefficient (C_0) = [FS] – (Floor([FS]/4))-1), Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

Expected Pre-cursor Coefficient = (Floor([FS]/4))-1

Expected Cursor Coefficient = [FS]-(Floor([FS]/4))-1)

Expected Post-cursor Coefficient = 0x00

Expected Reject Coefficient Values = 0

TEST CASE 37: Pre-cursor Coefficient (C_{-1}) = (Floor([FS]/4))-1, Cursor Coefficient (C_0) = [FS] – Floor([FS]/4), Post-cursor Coefficient (C_{+1}) = 0x01, Use Preset = 0

Expected Pre-cursor Coefficient = (Floor([FS]/4))-1

Expected Cursor Coefficient = [FS]-Floor([FS]/4)

Expected Post-cursor Coefficient = 0x01

Expected Reject Coefficient Values = 0 if [FS]-(2*Floor([FS]/4)) >= [LF]; 1 if [FS]-(2*Floor([FS]/4)) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 38: Pre-cursor Coefficient (C_{-1}) = 0x00, Cursor Coefficient (C_0) = [FS] – ((Floor([FS]/4))+1), Post-cursor Coefficient (C_{+1}) = (Floor([FS]/4))+1, Use Preset = 0

Expected Pre-cursor Coefficient = 0x00

Expected Cursor Coefficient = [FS]-((Floor([FS]/4))+1)

Expected Post-cursor Coefficient = (Floor([FS]/4))+1

Expected Reject Coefficient Values = 0 if [FS]-(2*Floor([FS]/4)) >= [LF]; 1 if [FS]-(2*Floor([FS]/4)) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 39 (only if [FS]-[LF] > 0): Pre-cursor Coefficient (C_{-1}) = 0x00, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = Floor(([FS]-[LF])/2), Use Preset = 0

Expected Pre-cursor Coefficient = 0x00

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = Floor(([FS]-[LF])/2)

Expected Reject Coefficient Values = 0

TEST CASE 40 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-1 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x01, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-1, Use Preset = 0

Expected Pre-cursor Coefficient = 0x01

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-1

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 41 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-2 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x02, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-2, Use Preset = 0

Expected Pre-cursor Coefficient = 0x02

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-2

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 42 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-3 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x03, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-3, Use Preset = 0

Expected Pre-cursor Coefficient = 0x03

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-3

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 43 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-4 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x04, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-4, Use Preset = 0

Expected Pre-cursor Coefficient = 0x04

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-4

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 44 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-5 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x05, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-5, Use Preset = 0

Expected Pre-cursor Coefficient = 0x05

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-5

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 45 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-6 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x06, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-6, Use Preset = 0

Expected Pre-cursor Coefficient = 0x06

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-6

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 46 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-7 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x07, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-7, Use Preset = 0

Expected Pre-cursor Coefficient = 0x07

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-7

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 47 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-8 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x08, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-8, Use Preset = 0

Expected Pre-cursor Coefficient = 0x08

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-8

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 48 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-9 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x09, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-9, Use Preset = 0

Expected Pre-cursor Coefficient = 0x09

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-9

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 49 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-10 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0A, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-10, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0A

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-10

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 50 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-11 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0B, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-11, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0B

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-11

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 51 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-12 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0C, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-12, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0C

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-12

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 52 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-13 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0D, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-13, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0D

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-13

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 53 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-14 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0E, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-14, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0E

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-14

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 54 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-15 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0F, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-15, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0F

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-15

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 55: Pre-cursor Coefficient (C_{-1}) = 0x3F, Cursor Coefficient (C_0) = 0x3F, Post-cursor Coefficient (C_{+1}) = 0x3F, Use Preset = 0

Expected Pre-cursor Coefficient = 0x3F

Expected Cursor Coefficient = 0x3F

Expected Post-cursor Coefficient = 0x3F

Expected Reject Coefficient Values = 1 (the coefficients add up to more than 63)

Note: Test Case 56 and the unpublished Test Cases 56A and 56B have been retracted. There is no harm in running these tests. If run, they do not affect the pass/fail status of the DUT.

The Expected Reject Coefficient Values for all Test Cases are derived based on the all the following rules combined:

i. FS is within the range {0x18,…,0x3F} (for full swing mode) or is within the range {0x0C,….,0x3F} (for reduced swing mode)

ii. C_0 = FS – C_{-1} – C_{+1} and is within the range {0x00,…,0x3F}

iii. C_{-1} = FS – C_0 – C_{+1} and is within the range {0x00,…, 0x3F}

iv. C_{+1} = FS – C_0 – C_{-1} and is within the range {0x00,…, 0x3F}

v. C_0 >= LF + C_{-1} + C_{+1}

vi. C_{-1} <= C_0 - C_{+1} - LF

vii. C_{+1} <= C_0 - C_{-1} - LF

viii. C_{-1} <= Floor(FS/4) (where Floor indicates rounding down to the nearest integer value and Ceiling indicates rounding up to the nearest integer value)

8. Step 7 is repeated until all Test Cases are tested, and then the PTC transitions to Phase 3. Note: The test must complete all Test Cases within 24 ms; otherwise the DUT may time out and exit link equalization.

9. In Phase 3 the PTC transmit TS1s with EC=11b, reflecting its current coefficient values, signaling the DUT to transition to Phase 3. If the PTC receives 2 consecutive TS1s with EC = 11b and requesting a valid new preset or coefficient, it processes these normally. If the PTC receives any preset request in the range of 0xB to 0xF the PTC will reject this (Reject Coefficient

is set to 1 in the PTC's response) and the DUT will not fail the test and testing will continue. If the PTC receives any coefficient request that is illegal for its advertised FS, LF range the PTC will reject this (Reject Coefficient is set to 1 in the PTC's response) and the DUT will fail the test. Otherwise when the PTC receives 2 consecutive TS1 with EC = 00b, the PTC transitions to the Recovery.RcvrLock state, then the PTC goes to electrical idle and times out the DUT and ends the test. If the PTC does not receive either 2 consecutive TS1s with EC = 11b or 2 consecutive TS1s with EC = 00b, within 32 ms, the DUT fails.

10. If at any time in the above steps, for a Test Case where the Expected Rejected Coefficient Value is 0 and the PTC link loses contact with the DUT, that Test Case is skipped. (This may be the result of a legal preset that is electrically incompatible with the channel.) However, at least one of these Test Cases must not lose contact in order for the DUT to pass the test.

    If at any time in the above steps, for a Test Case where the Expected Reject Coefficient Values is 1 and the PTC link loses contact with the DUT, the DUT fails if the previous Test Case did not lose contact with the DUT, and the Test Case is skipped if the previous Test Case also lost contact with the DUT.

11. If all the conditions above are met, then the DUT passes the test.

12. If any of the conditions above are not met (excluding losing contact with the DUT pursuant to step 10, or warnings), log it as DUT's failure.

3.7.3.2 DUT is an Add-in Card or an Upstream Port of a Switch

1. Perform steps given in Section A.2.2 to reach Recovery.RcvrLock.
2. The PTC transmits TS1s with non-PAD link and lane numbers set during the Configuration states. The speed_change bit is set to 1. All data rates are advertised. The PTC shall transmit EQ TS1 (symbol 6, bit 7 set to 1) with transmitter preset values (symbol 6, bits 6:3) of 0x0 and receiver preset hint values (symbol 6, bits 2:0) of 0x0. All data rates are advertised. After receiving 8 consecutive TS1 or 8 consecutive TS2 with identical link and lane numbers to the one being transmitted the PTC transitions to Recovery.RcvrCfg.
3. The PTC transmits TS2s with non-PAD link and lane numbers. All data rates are advertised. The PTC shall transmit EQ TS2 (symbol 6, bit 7 set to 1) with transmitter preset values (symbol 6, bits 6:3) of 0x0 and receiver preset hint values (symbol 6, bits 2:0) of 0x0. All data rates are advertised. After receiving 8 consecutive TS2s with speed_change bit set to 1 and 8.0 GT/s advertised, and after transmitting 32 TS2s with speed_change bit set to 1 and all data rates advertised after receiving the first TS2 and without interruption by an EIEOS, the PTC transitions to Recovery.Speed. If the transmission of 32 TS2s is interrupted by an EIEOS, the counting shall be reset to 0 and the 32 TS2s shall be retransmitted. If the received TS2s do not advertise 8.0 GT/s, then the test case is aborted, and the test result is reported as skipped.
4. In Recovery.Speed the PTC transmits 1 EIOS and goes to electrical idle. The PTC shall change to 8.0 GT/s data rate within 0.5 ms. The next state is Recovery.RcvrLock.
5. The PTC enters Phase 1 of the Recovery.Equalization state whereas the DUT enters Phase 0. The PTC transmits TS1s with EC = 01b, Tx equalization sets the presets of 0x0, and reflects its current FS, LF, and post-coefficient values. The PTC should first see the DUT sending TS1s with EC = 00b reflecting the preset value it has requested in the EQ TS2s. Then the PTC

should then see the DUT sending TS1s with EC = 01b. The PTC should record per lane the FS (symbol 7, bits 5-0) and LF (symbol 8, bits 5-0) values received in two consecutive TS1s with EC = 01b (to be used later in Phase 3). If the recorded LF value is greater than the FS value on any lane, the DUT fails. After receiving 2 consecutive TS1s with EC = 01b within 2 ms the PTC transitions to Phase 2. If the PTC does not receive 2 consecutive TS1s with EC = 01b within 24 ms, the DUT fails.

6. In Phase 2, the PTC transmits TS1s with EC = 10b, reflecting its current coefficient values. If the PTC receives 2 consecutive TS1s with EC = 10b and requesting a valid new preset or coefficient, it processes these normally. If the PTC receives any preset request in the range of 0xB to 0xF the PTC will reject this (Reject Coefficient is set to 1 in the PTC's response) and the DUT will not fail the test and testing will continue. If the PTC receives any coefficient request that is illegal for its advertised FS, LF range the PTC will reject this (Reject Coefficient is set to 1 in the PTC's response) and the DUT will fail the test. Otherwise, when the PTC receives 2 consecutive TS2 with EC = 11b, the PTC transitions to Phase 3. If the PTC does not receive 2 consecutive TS1s with EC = 11b within 32 ms, the DUT fails.
7. In Phase 3, the PTC transmits TS1s with EC=11b, with the Use Preset bit (symbol 6, bit 7) set to a Test Case value (starting at Test Case 1), and transmitter pre-cursor/cursor/post-cursor coefficient values (symbol 7, bits 5:0/symbol 8, bits 5:0/symbol 9, bits 5:0) from a Test Case (starting at Test Case 1). The PTC must transmit these TS1 for at least 1 µs. The PTC shall obtain new block alignment every time it receives the new coefficients. The PTC should wait at least 1 µs and then check that the received TS1 contains the expected Reject Coefficient Values, Pre-cursor, Cursor, and Post-cursor coefficient values (symbol 9 bit 6, symbol 7 bits 5:0, symbol 8 bits 5:0, and symbol 9 bits 5:0 respectively) for the Test Case (starting at Test Case 1). If this check fails and the Test Case's Expected Reject Coefficient Values is 0, the DUT fails the test. If this check fails and the Test Case's Expected Reject Coefficient Values is 1, the DUT is issued a warning. The range of valid values is given by the following (where [FS] and [LF] are the values captured in step 5):

   TEST CASE 1: Pre-cursor Coefficient (C_{-1}) = 0x00, Cursor Coefficient (C_0) = 0x00, Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

   Expected Pre-cursor Coefficient = 0x00

   Expected Cursor Coefficient = 0x00

   Expected Post-cursor Coefficient = 0x00

   Expected Reject Coefficient Values = 1 (the coefficients add up to less than 12)

   TEST CASE 2: Pre-cursor Coefficient (C_{-1}) = 0x00, Cursor Coefficient (C_0) = 0x0B, Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

   Expected Pre-cursor Coefficient = 0x00

   Expected Cursor Coefficient = 0x0B

   Expected Post-cursor Coefficient = 0x00

   Expected Reject Coefficient Values = 1 (the coefficients add up to less than 12)

TEST CASE 3: Pre-cursor Coefficient (C_{-1}) = 0x0B, Cursor Coefficient (C_0) = 0x00, Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0B

Expected Cursor Coefficient = 0x00

Expected Post-cursor Coefficient = 0x00

Expected Reject Coefficient Values = 1 (the coefficients add up to less than 12)

TEST CASE 4: Pre-cursor Coefficient (C_{-1}) = 0x00, Cursor Coefficient (C_0) = 0x00, Post-cursor Coefficient (C_{+1}) = 0x0B, Use Preset = 0

Expected Pre-cursor Coefficient = 0x00

Expected Cursor Coefficient = 0x00

Expected Post-cursor Coefficient = 0x0B

Expected Reject Coefficient Values = 1 (the coefficients add up to less than 12)

TEST CASE 5: Pre-cursor Coefficient (C_{-1}) = 0x01, Cursor Coefficient (C_0) = 0x09, Post-cursor Coefficient (C_{+1}) = 0x01, Use Preset = 0

Expected Pre-cursor Coefficient = 0x01

Expected Cursor Coefficient = 0x09

Expected Post-cursor Coefficient = 0x01

Expected Reject Coefficient Values = 1 (the coefficients add up to less than 12)

TEST CASE 6: Pre-cursor Coefficient (C_{-1}) = 0x00, Cursor Coefficient (C_0) = [FS], Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

Expected Pre-cursor Coefficient = 0x00

Expected Cursor Coefficient = [FS]

Expected Post-cursor Coefficient = 0x00

Expected Reject Coefficient Values = 0

TEST CASE 7 (only if [FS] < 63): Pre-cursor Coefficient (C_{-1}) = 0x00, Cursor Coefficient (C_0) = [FS]+1, Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

Expected Pre-cursor Coefficient = 0x00

Expected Cursor Coefficient = [FS]+1

Expected Post-cursor Coefficient = 0x00

Expected Reject Coefficient Values = 1 (the coefficients add up to more than [FS])

TEST CASE 8: Pre-cursor Coefficient (C_{-1}) = 0x00, Cursor Coefficient (C_0) = [FS]-1, Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

Expected Pre-cursor Coefficient = 0x00

Expected Cursor Coefficient = [FS]-1

Expected Post-cursor Coefficient = 0x00

Expected Reject Coefficient Values = 1 (the coefficients add up to less than [FS])

TEST CASE 9: Pre-cursor Coefficient (C_{-1}) = 0x01, Cursor Coefficient (C_0) = [FS], Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

Expected Pre-cursor Coefficient = 0x01

Expected Cursor Coefficient = [FS]

Expected Post-cursor Coefficient = 0x00

Expected Reject Coefficient Values = 1 (the coefficients add up to more than [FS])

TEST CASE 10: Pre-cursor Coefficient (C_{-1}) = 0x00, Cursor Coefficient (C_0) = [FS], Post-cursor Coefficient (C_{+1}) = 0x01, Use Preset = 0

Expected Pre-cursor Coefficient = 0x00

Expected Cursor Coefficient = [FS]

Expected Post-cursor Coefficient = 0x01

Expected Reject Coefficient Values = 1 (the coefficients add up to more than [FS])

TEST CASE 11: Pre-cursor Coefficient (C_{-1}) = 0x01, Cursor Coefficient (C_0) = [FS]-1, Post-cursor Coefficient (C_{+1}) = 0x01, Use Preset = 0

Expected Pre-cursor Coefficient = 0x01

Expected Cursor Coefficient = [FS]-1

Expected Post-cursor Coefficient = 0x01

Expected Reject Coefficient Values = 1 (the coefficients add up to more than [FS])

TEST CASE 12: Pre-cursor Coefficient (C_{-1}) = 0x01, Cursor Coefficient (C_0) = [FS]-2, Post-cursor Coefficient (C_{+1}) = 0x01, Use Preset = 0

Expected Pre-cursor Coefficient = 0x01

Expected Cursor Coefficient = [FS]-2

Expected Post-cursor Coefficient = 0x01

Expected Reject Coefficient Values = 0 if ([FS]-4) >= [LF]; 1 if ([FS]-4) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 13: Pre-cursor Coefficient (C_{-1}) = 0x01, Cursor Coefficient (C_0) = [FS]-3, Post-cursor Coefficient (C_{+1}) = 0x01, Use Preset = 0

Expected Pre-cursor Coefficient = 0x01

Expected Cursor Coefficient = [FS]-3

Expected Post-cursor Coefficient = 0x01

Expected Reject Coefficient Values = 1 (the coefficients add up to less than [FS])

TEST CASE 14: Pre-cursor Coefficient (C_{-1}) = 0x02, Cursor Coefficient (C_0) = [FS]-4, Post-cursor Coefficient (C_{+1}) = 0x02, Use Preset = 0

Expected Pre-cursor Coefficient = 0x02

Expected Cursor Coefficient = [FS]-4

Expected Post-cursor Coefficient = 0x02

Expected Reject Coefficient Values = 0 if ([FS]-8) >= [LF]; 1 if ([FS]-8) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 15: Pre-cursor Coefficient (C_{-1}) = 0x03, Cursor Coefficient (C_0) = [FS]-6, Post-cursor Coefficient (C_{+1}) = 0x03, Use Preset = 0

Expected Pre-cursor Coefficient = 0x03

Expected Cursor Coefficient = [FS]-6

Expected Post-cursor Coefficient = 0x03

Expected Reject Coefficient Values = 0 if ([FS]-12) >= [LF]; 1 if ([FS]-12) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 16: Pre-cursor Coefficient (C_{-1}) = 0x04, Cursor Coefficient (C_0) = [FS]-8, Post-cursor Coefficient (C_{+1}) = 0x04, Use Preset = 0

Expected Pre-cursor Coefficient = 0x04

Expected Cursor Coefficient = [FS]-8

Expected Post-cursor Coefficient = 0x04

Expected Reject Coefficient Values = 0 if ([FS]-16) >= [LF]; 1 if ([FS]-16) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 17: Pre-cursor Coefficient (C_{-1}) = 0x05, Cursor Coefficient (C_0) = [FS]-10, Post-cursor Coefficient (C_{+1}) = 0x05, Use Preset = 0

Expected Pre-cursor Coefficient = 0x05

Expected Cursor Coefficient = [FS]-10

Expected Post-cursor Coefficient = 0x05

Expected Reject Coefficient Values = 0 if ([FS]-20) >= [LF]; 1 if ([FS]-20) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 18: Pre-cursor Coefficient (C_{-1}) = 0x06, Cursor Coefficient (C_0) = [FS]-12, Post-cursor Coefficient (C_{+1}) = 0x06, Use Preset = 0

Expected Pre-cursor Coefficient = 0x06

Expected Cursor Coefficient = [FS]-12

Expected Post-cursor Coefficient = 0x06

Expected Reject Coefficient Values = 0 if ([FS]-24) >= [LF]; 1 if ([FS]-24) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 19 (only if [FS]-14 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x07, Cursor Coefficient (C_0) = [FS]-14, Post-cursor Coefficient (C_{+1}) = 0x07, Use Preset = 0

Expected Pre-cursor Coefficient = 0x07

Expected Cursor Coefficient = [FS]-14

Expected Post-cursor Coefficient = 0x07

Expected Reject Coefficient Values = 0 if ([FS]-28) >= [LF]; 1 if ([FS]-28) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 20 (only if [FS]-16 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x08, Cursor Coefficient (C_0) = [FS]-16, Post-cursor Coefficient (C_{+1}) = 0x08, Use Preset = 0

Expected Pre-cursor Coefficient = 0x08

Expected Cursor Coefficient = [FS]-16

Expected Post-cursor Coefficient = 0x08

Expected Reject Coefficient Values = 0 if ([FS]-32) >= [LF]; 1 if ([FS]-32) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 21 (only if [FS]-18 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x09, Cursor Coefficient (C_0) = [FS]-18, Post-cursor Coefficient (C_{+1}) = 0x09, Use Preset = 0

Expected Pre-cursor Coefficient = 0x09

Expected Cursor Coefficient = [FS]-18

Expected Post-cursor Coefficient = 0x09

Expected Reject Coefficient Values = 0 if ([FS]-36) >= [LF]; 1 if ([FS]-36) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 22 (only if [FS]-20 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0A, Cursor Coefficient (C_0) = [FS]-20, Post-cursor Coefficient (C_{+1}) = 0x0A, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0A

Expected Cursor Coefficient = [FS]-20

Expected Post-cursor Coefficient = 0x0A

Expected Reject Coefficient Values = 0 if ([FS]-40) >= [LF]; 1 if ([FS]-40) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 23 (only if [FS]-22 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0B, Cursor Coefficient (C_0) = [FS]-22, Post-cursor Coefficient (C_{+1}) = 0x0B, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0B

Expected Cursor Coefficient = [FS]-22

Expected Post-cursor Coefficient = 0x0B

Expected Reject Coefficient Values = 0 if ([FS]-44) >= [LF]; 1 if ([FS]-44) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 24 (only if [FS]-24 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0C, Cursor Coefficient (C_0) = [FS]-24, Post-cursor Coefficient (C_{+1}) = 0x0C, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0C

Expected Cursor Coefficient = [FS]-24

Expected Post-cursor Coefficient = 0x0C

Expected Reject Coefficient Values = 0 if ([FS]-48) >= [LF]; 1 if ([FS]-48) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 25 (only if [FS]-26 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0D, Cursor Coefficient (C_0) = [FS]-26, Post-cursor Coefficient (C_{+1}) = 0x0D, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0D

Expected Cursor Coefficient = [FS]-26

Expected Post-cursor Coefficient = 0x0D

Expected Reject Coefficient Values = 0 if ([FS]-52) >= [LF]; 1 if ([FS]-52) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 26 (only if [FS]-28 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0E, Cursor Coefficient (C_0) = [FS]-28, Post-cursor Coefficient (C_{+1}) = 0x0E, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0E

Expected Cursor Coefficient = [FS]-28

Expected Post-cursor Coefficient = 0x0E

Expected Reject Coefficient Values = 0 if ([FS]-56) >= [LF]; 1 if ([FS]-56) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 27 (only if [FS]-30 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0F, Cursor Coefficient (C_0) = [FS]-30, Post-cursor Coefficient (C_{+1}) = 0x0F, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0F

Expected Cursor Coefficient = [FS]-30

Expected Post-cursor Coefficient = 0x0F

Expected Reject Coefficient Values = 0 if ([FS]-60) >= [LF]; 1 if ([FS]-60) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 28 (only if [FS] > [LF]): Pre-cursor Coefficient (C_{-1}) = 0x00, Cursor Coefficient (C_0) = [LF], Post-cursor Coefficient (C_{+1}) = [FS] - [LF], Use Preset = 0

Expected Pre-cursor Coefficient = 0x00

Expected Cursor Coefficient = [LF]

Expected Post-cursor Coefficient = [FS]-[LF]

Expected Reject Coefficient Values = 1 (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 29 (only if [FS] > [LF]): Pre-cursor Coefficient (C_{-1}) = [FS] - [LF], Cursor Coefficient (C_0) = [LF], Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

Expected Pre-cursor Coefficient = [FS]-[LF]

Expected Cursor Coefficient = [LF]

Expected Post-cursor Coefficient = 0x00

Expected Reject Coefficient Values = 1 (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 30 (only if [FS] > [LF]): Pre-cursor Coefficient (C_{-1}) = Floor(([FS] - [LF])/2(, Cursor Coefficient (C_0) = [LF], Post-cursor Coefficient (C_{+1}) = Ceiling(([FS] - [LF])/2), Use Preset = 0

Expected Pre-cursor Coefficient = Floor(([FS]-[LF])/2)

Expected Cursor Coefficient = [LF]

Expected Post-cursor Coefficient = Ceiling(([FS]-[LF])/2)

Expected Reject Coefficient Values = 1 (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 31: Pre-cursor Coefficient (C_{-1}) = 0x00, Cursor Coefficient (C_0) = [LF]-1, Post-cursor Coefficient (C_{+1}) = [FS]-[LF]+1, Use Preset = 0

Expected Pre-cursor Coefficient = 0x00

Expected Cursor Coefficient = [LF]-1

Expected Post-cursor Coefficient = [FS]-[LF]+1

Expected Reject Coefficient Values = 1 (the post-cursor coefficient is greater than the cursor coefficient minus [LF])

TEST CASE 32: Pre-cursor Coefficient (C_{-1}) = [FS]-[LF]+1, Cursor Coefficient (C_0) = [LF]-1, Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

Expected Pre-cursor Coefficient = [FS]-[LF]+1

Expected Cursor Coefficient = [LF]-1

Expected Post-cursor Coefficient = 0x00

Expected Reject Coefficient Values = 1 (the pre-cursor coefficient is greater than the cursor coefficient minus [LF])

TEST CASE 33 (only if [FS] > [LF]): Pre-cursor Coefficient (C_{-1}) = Floor(([FS]-[LF])/2)+1, Cursor Coefficient (C_0) = [LF]+1, Post-cursor Coefficient (C_{+1}) = Ceiling(([FS]-[LF])/2)+1, Use Preset = 0

Expected Pre-cursor Coefficient = Floor(([FS]-[LF])/2)+1

Expected Cursor Coefficient = [LF]+1

Expected Post-cursor Coefficient = Ceiling(([FS]-[LF])/2)+1

Expected Reject Coefficient Values = 1 (the coefficients add up to more than [FS])

TEST CASE 34: Pre-cursor Coefficient (C_{-1}) = Floor([FS]/4), Cursor Coefficient (C_0) = [FS] – Floor([FS]/4), Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

Expected Pre-cursor Coefficient = Floor([FS]/4)

Expected Cursor Coefficient = [FS]-Floor([FS]/4)

Expected Post-cursor Coefficient = 0x00

Expected Reject Coefficient Values = 0

TEST CASE 35: Pre-cursor Coefficient (C_{-1}) = (Floor([FS]/4))+1, Cursor Coefficient (C_0) = [FS] – ((Floor([FS]/4))+1), Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

Expected Pre-cursor Coefficient = (Floor([FS]/4))+1

Expected Cursor Coefficient = [FS]-((Floor([FS]/4))+1)

Expected Post-cursor Coefficient = 0x00

Expected Reject Coefficient Values = 1 (the pre-cursor coefficient is greater than [FS]/4)

TEST CASE 36: Pre-cursor Coefficient (C_{-1}) = (Floor([FS]/4))-1, Cursor Coefficient (C_0) = [FS] – (Floor([FS]/4))-1), Post-cursor Coefficient (C_{+1}) = 0x00, Use Preset = 0

Expected Pre-cursor Coefficient = (Floor([FS]/4))-1

Expected Cursor Coefficient = [FS]-(Floor([FS]/4))-1)

Expected Post-cursor Coefficient = 0x00

Expected Reject Coefficient Values = 0

TEST CASE 37: Pre-cursor Coefficient (C_{-1}) = (Floor([FS]/4))-1, Cursor Coefficient (C_0) = [FS] – Floor([FS]/4), Post-cursor Coefficient (C_{+1}) = 0x01, Use Preset = 0

Expected Pre-cursor Coefficient = (Floor([FS]/4))-1

Expected Cursor Coefficient = [FS]-Floor([FS]/4)

Expected Post-cursor Coefficient = 0x01

Expected Reject Coefficient Values = 0 if [FS]-(2*Floor([FS]/4)) >= [LF]; 1 if [FS]-(2*Floor([FS]/4)) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 38: Pre-cursor Coefficient (C_{-1}) = 0x00, Cursor Coefficient (C_0) = [FS] – ((Floor([FS]/4))+1), Post-cursor Coefficient (C_{+1}) = (Floor([FS]/4))+1, Use Preset = 0

Expected Pre-cursor Coefficient = 0x00

Expected Cursor Coefficient = [FS]-((Floor([FS]/4))+1)

Expected Post-cursor Coefficient = (Floor([FS]/4))+1

Expected Reject Coefficient Values = 0 if [FS]-(2*Floor([FS]/4)) >= [LF]; 1 if [FS]-(2*Floor([FS]/4)) < [LF] (cursor coefficient minus both pre-cursor and post-cursor coefficients must be greater than or equal to [LF])

TEST CASE 39 (only if [FS]-[LF] > 0): Pre-cursor Coefficient (C_{-1}) = 0x00, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = Floor(([FS]-[LF])/2), Use Preset = 0

Expected Pre-cursor Coefficient = 0x00

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = Floor(([FS]-[LF])/2)

Expected Reject Coefficient Values = 0

TEST CASE 40 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-1 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x01, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-1, Use Preset = 0

Expected Pre-cursor Coefficient = 0x01

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-1

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 41 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-2 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x02, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-2, Use Preset = 0

Expected Pre-cursor Coefficient = 0x02

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-2

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 42 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-3 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x03, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-3, Use Preset = 0

Expected Pre-cursor Coefficient = 0x03

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-3

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 43 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-4 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x04, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-4, Use Preset = 0

Expected Pre-cursor Coefficient = 0x04

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-4

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 44 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-5 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x05, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-5, Use Preset = 0

Expected Pre-cursor Coefficient = 0x05

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-5

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 45 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-6 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x06, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-6, Use Preset = 0

Expected Pre-cursor Coefficient = 0x06

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-6

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 46 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-7 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x07, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-7, Use Preset = 0

Expected Pre-cursor Coefficient = 0x07

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-7

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 47 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-8 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x08, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-8, Use Preset = 0

Expected Pre-cursor Coefficient = 0x08

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-8

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 48 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-9 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x09, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-9, Use Preset = 0

Expected Pre-cursor Coefficient = 0x09

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-9

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 49 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-10 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0A, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-10, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0A

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-10

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 50 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-11 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0B, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-11, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0B

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-11

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 51 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-12 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0C, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-12, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0C

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-12

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 52 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-13 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0D, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-13, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0D

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-13

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 53 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-14 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0E, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-14, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0E

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-14

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 54 (only if [FS]-[LF] > 0 and Floor (([FS]-[LF])/2)-15 >= 0): Pre-cursor Coefficient (C_{-1}) = 0x0F, Cursor Coefficient (C_0) = Ceiling(([FS]-[LF])/2)+[LF], Post-cursor Coefficient (C_{+1}) = (Floor(([FS]-[LF])/2))-15, Use Preset = 0

Expected Pre-cursor Coefficient = 0x0F

Expected Cursor Coefficient = Ceiling(([FS]-[LF])/2)+[LF]

Expected Post-cursor Coefficient = (Floor(([FS]-[LF])/2))-15

Expected Reject Coefficient Values = 0 if Floor([FS]/4) >= C_{-1}; 1 Floor([FS]/4) < C_{-1} (pre-cursor coefficient must be less than Floor[FS]/4)

TEST CASE 55: Pre-cursor Coefficient (C_{-1}) = 0x3F, Cursor Coefficient (C_0) = 0x3F, Post-cursor Coefficient (C_{+1}) = 0x3F, Use Preset = 0

Expected Pre-cursor Coefficient = 0x3F

Expected Cursor Coefficient = 0x3F

Expected Post-cursor Coefficient = 0x3F

Expected Reject Coefficient Values = 1 (the coefficients add up to more than 63)

Note: Test Case 56 and the unpublished Test Cases 56A and 56B have been retracted. There is no harm in running these tests. If run, they do not affect the pass/fail status of the DUT.

The Expected Reject Coefficient Values for all Test Cases are derived based on the all the following rules combined:

ix. FS is within the range {0x18,…,0x3F} (for full swing mode) or is within the range {0x0C,….,0x3F} (for reduced swing mode)

x. $C_0 = FS – C_{-1} – C_{+1}$ and is within the range {0x00,…,0x3F}

xi. $C_{-1} = FS – C_0 – C_{+1}$ and is within the range {0x00,…, 0x3F}

xii. $C_{+1} = FS – C_0 – C_{-1}$ and is within the range {0x00,…, 0x3F}

xiii. $C_0 >= LF + C_{-1} + C_{+1}$

xiv. $C_{-1} <= C_0 - C_{+1} - LF$

xv. $C_{+1} <= C_0 - C_{-1} - LF$

xvi. $C_{-1} <= Floor(FS/4)$ (where Floor indicates rounding down to the nearest integer value and Ceiling indicates rounding up to the nearest integer value)

8. Step 7 is repeated until all Test Cases are tested, and then the PTC transitions to the Recovery.RcvrLock state, then the PTC goes to electrical idle and times out the DUT and ends the test. Note: The test must complete all Test Cases within 24 ms; otherwise the DUT may time out and exit link equalization.

9. If at any time in the above steps, for a Test Case where the Expected Reject Coefficient Values is 0 and the PTC link loses contact with the DUT, that Test Case is skipped. (This may be the result of a legal preset that is electrically incompatible with the channel.) However, at least one of the Test Cases must not lose contact in order for these DUT to pass the test.

   If at any time in the above steps, for a Test Case where the Expected Reject Coefficient Values is 1 and the PTC link loses contact with the DUT, the DUT fails if the previous Test Case did not lose contact with the DUT, and the Test Case is skipped if the previous Test Case also lost contact with the DUT.

10. If all the conditions above are met, then the DUT passes the test.

11. If any of the conditions above are not met (excluding losing contact with the DUT pursuant to step 9, or warnings), log it as DUT's failure.

3.8 Loopback (Informative)

These tests verify that the DUT correctly enters Loopback when a standard sequence is used (i.e., a sequence that is expected to be used by test equipment to perform receiver testing). In the Loopback state the DUT is supposed to just transmit the data it receives without any modification except for the SKP ordered sets.

There are two ways a DUT can enter Loopback. One is through the Configuration.Linkwidth.Start substate and the other one is through the Recovery.Idle state. If Link Equalization is to be performed at 8.0 GT/s before going to Loopback, then only the path through Recovery.Idle shall be used. For testing with a fixed preset the path through Configuration.Linkwidth.Start shall be used.

3.8.1 Test 60-10 Loopback through Configuration.Linkwidth.Start

1. The PTC starts in the Detect.Quiet state and waits for 12 ms and then transitions to Detect.Active.

2. In Detect.Active PTC performs receiver detection, on detecting a receiver it transitions to Polling.Active, otherwise it shall go back to Detect.Quiet, and if it does not detect a receiver within 1 s, the test fails.

   In Polling.Active state the PTC transmits TS1s with link and lane numbers set to PAD, the Compliance Receive bit set to 0, Hot Reset bit set to 0, Disable Link bit set to 0, and Loopback bit set to 1. The data rate that is to be used in Loopback state shall be advertised based on the expected test result, as the link will enter Loopback at the highest negotiated link speed. After transmitting TS1s for 2 ms the PTC transitions to Polling.Configuration.

   TEST CASE 1: Advertised Data Rates = 2.5 GT/s

   TEST CASE 2: (For PCIe2.x or later testing) Advertised Data Rates = 2.5 GT/s and 5.0 GT/s

   TEST CASE 3: (For PCIe3.x or later testing) Advertised Data Rates = 2.5 GT/s and 5.0 GT/s and 8.0 GT/s

3. In Polling.Configuration the PTC transmits TS2s with Hot Reset bit set to 0, Disable Link bit set to 0, and Loopback bit set to 1 and link and lane numbers set to PAD. The data rate that is to be used in Loopback state shall be advertised based on the expected test result, as the link will enter Loopback at the highest negotiated link speed. After transmitting TS2s for 10 ms the PTC transitions to Configuration.Linkwidth.Start.

4. From Configuration.Linkwidth.Start the PTC enters Loopback.Entry and transmits TS1s (Configuration.Linkwidth.Start is just a bypass state). The PTC is the loopback master and the DUT is the loopback slave. If 8.0 GT/s is going to be the speed of operation during Loopback, the PTC can choose to transmit presets using the EQ TS1. If not, the DUT will start the 8.0 GT/s using its own default settings.

5. If the current speed of operation is not the highest common speed then the PTC shall transmit 16 consecutive TS1s with Hot Reset bit set to 0, Disable Link bit set to 0, and Loopback bit set to 1, Selectable De-emphasis bit set to the desired value. If the highest common data rate is 8.0 GT/s, these TS1s are EQ TS1s with the desired Preset (0x0).

6. The PTC shall transmit the proper number of EIOS (one EIOS for 2.5 GT/s; two EIOS for 5.0 GT/s; one EIOS for 8.0 GT/s) and then goes to electrical idle for 1 ms. The PTC shall change the speed of operation to the highest common advertised date rate during this 1 ms.

7. After coming out of electrical idle, the PTC transmits TS1s with Compliance Receive bit set to 0, Hot Reset bit set to 0, Disable Link bit set to 0, and Loopback bit set, at the new data rate. For 2.5 GT/s and 5.0 GT/s, 8b/10b encoding is used and for 8.0 GT/s, 128b/130b encoding is used. If 8.0 GT/s is going to be the speed of operation during loopback, the PTC can choose to transmit presets using the EQ TS1. If not, the DUT will start the 8.0 GT/s using its own default settings.

8. When the PTC receives two consecutive TS1s with Loopback bit set to 1 in the PTC transitions to Loopback.Active. If the PTC does not receive two consecutive TS1s with the Loopback bit set to 1 within 100 ms, the test ends and reports failure.

9. The PTC starts sending the test pattern. The DUT just replicates the data it receives and transmits it back to the PTC. The PTC checks that the received data matches the transmitted data (except for SKPs).

Test Pattern

10. The PTC sends SKP ordered sets properly (one SKP every 1538 symbol times for 8b/10b encoding, two SKPs every 375 blocks for 128b/130b encoding. The DUT can either add or subtract SKP symbols, but must only add SKPs adjacent to received SKPs.

11. The PTC completes sending the test pattern and then sends 4 consecutive EIOS, then transitions to Loopback.Exit.

12. In Loopback.Exit, the PTC transmits the proper number of EIOS (one EIOS for 2.5 GT/s, 8 EIOS for 5.0 GT/s and 8.0 GT/s) and then enters electrical idle for 2 ms. The PTC checks that the DUT sends the same number of EIOS back. The PTC checks that the DUT then enters electrical idle.

13. The PTC then goes to Detect.

3.8.2 Test 60-20 Loopback Through Recovery.Idle

3.8.2.1 DUT is a Motherboard or a Downstream port of a Switch or Bridge

1. Perform steps given in Section A.2.2 to reach Recovery.RcvrLock.

2. The PTC transmits TS1s with non-PAD link and lane numbers set during the Configuration states. The speed_change bit is set to 1. The data rates that are advertised are selected from the test case (starting at the first test case). After receiving 8 consecutive TS1s or 8 consecutive TS2s with identical link and lane numbers to the ones being transmitted the PTC transitions to Recovery.RcvrCfg.

   TEST CASE 1: Advertised Data Rates = 2.5 GT/s

   TEST CASE 2: (For PCIe2.x or later testing) Advertised Data Rates = 2.5 GT/s and 5.0 GT/s

   TEST CASE 3: (For PCIe3.x or later testing) Advertised Data Rates = 2.5 GT/s and 5.0 GT/s and 8.0 GT/s

3. The PTC transmits TS2s with non-PAD link and lane numbers. If 8.0 GT/s is going to be the data rate of operation during loopback the PTC shall let the DUT start 8.0 GT/s using its own default Tx preset settings and not send any EQ TS1 or EQ TS2 preset requests. After receiving 8 consecutive TS2s with speed_change bit set to 1, and after transmitting 32 TS2s with speed_change bit set to 1 after receiving the first TS2 and without interruption by an EIEOS, the PTC transitions to Recovery.Speed. If the transmission of 32 TS2s is interrupted by an EIEOS, then the counting shall be reset to 0 and 32 TS2s shall be retransmitted.

4. In Recovery.Speed the PTC transmits 1 EIOS and goes to electrical idle. The PTC changes to the highest common advertised data rate 0.5 ms after transmitting the EIOS. The next state is Recovery.RcvrLock.

5. If the operating data rate is 8.0 GT/s, then the PTC transitions to Recovery.Equalization from Recovery.RcvrLock and Link Equalization is performed. If the operating data rate is 2.5 GT/s or 5.0 GT/s, then the PTC stays in Recovery.RcvrLock and skips to step 10.

6. The PTC enters Phase 0. The PTC transmits TS1s with EC = 00b, Tx equalization sets the presets it received in the EQ TS2s, and reflects its current coefficient values. After receiving 2 consecutive TS1s with EC = 01b within 2 ms, the PTC transitions to Phase 1. If 2 consecutive TS1s as described above are not received within 2 ms, the test is considered to be failing and the link falls back to 2.5 GT/s. Alternately, if the PTC receives 8 consecutive TS1s with EC = 00b, the PTC goes to the Recovery.RcvrLock state and ends the test (this is not a failure, but instead the test result is reported as skipped). If the PTC does not receive either 2 consecutive TS1s with EC = 01b or 8 consecutive TS1s with EC = 00b, within 12 ms, the DUT fails.

7. In Phase 1, the PTC transmits TS1s with EC=01b, with preset set to the one requested in the EQ TS2s it received from the DUT and reflects its current FS, LF, and post-coefficient values. After receiving 2 consecutive TS1s with EC = 10b within 2 ms it transitions to Phase 2. Alternately, if the PTC receives 8 consecutive TS1s with EC = 00b, the PTC goes to the Recovery.RcvrLock state and ends the test (this is not a failure, but instead the test result is reported as skipped). If the PTC does not receive either 2 consecutive TS1s with EC = 10b or 8 consecutive TS1s with EC = 00b, within 12 ms, the DUT fails.

8. In Phase 2 the PTC transmits TS1 with EC = 10b, with the Use Preset bit (symbol 6, bit 7) set to 1, and transmitter preset values (symbol 6, bits 6:3) set to 0x0. The PTC must transmit these TS1 for at least 1 μs. The PTC shall obtain new block alignment every time it receives the new coefficients. The PTC should check that the new preset is being reflected within (500 + roundtrip time + logic delay) ns in the received TS1 (symbol 6, bits 6:3) and it matches the expected 0x0 value, and also should verify that the Reject Coefficient Values bit (symbol 9, bit 6) matches the expected 0 value. If not, the DUT is fails the test. The PTC shall then transition to Phase 3.

9. In Phase 3 the PTC transmit TS1s with EC = 11b, reflecting its current coefficient values, signaling the DUT to transition to Phase 3. If the PTC receives 2 consecutive TS1s with EC = 11b and requesting a new preset or coefficient, it processes these normally by either accepting or rejecting it as required (a rejected preset or coefficient is not a failure). When the PTC receives 2 consecutive TS1s with EC = 00b the PTC transitions to Recovery.RcvrLock.

10. In Recovery.RcvrLock the PTC transmits TS1s with speed_change bit set to 0. After receiving 8 consecutive TS1 or 8 consecutive TS2 with identical link and lane numbers the PTC transitions to Recovery.RcvrCfg.

11. In Recovery.RcvrCfg the PTC transmits TS2s. After receiving 8 consecutive TS2s with speed_change bit set to 0 and at least 16 consecutive TS2s have been transmitted after receiving the first TS2, the PTC transitions to Loopback.Entry via Recovery.Idle (which is just a pass through state).

12. In Loopback.Entry the PTC transmits TS1s with Compliance Receive bit set to 0, Hot Reset bit set to 0, Disable Link bit set to 0, and Loopback bit set to 1. When the PTC receives 2 consecutive TS1s with the Loopback bit set to 1, the DUT is in Loopback.Active. If it does not see 2 consecutive TS1s with the Loopback bit set to 1 from the DUT within 2 ms, the test fails.

13. The PTC starts sending the test pattern. The DUT must replicate the data it receives and transmits it back to the PTC. The PTC checks that the received data matches the transmitted data (except for SKPs). The PTC sends SKP ordered sets properly (one SKP Ordered Set every 1538 symbol times for 8b/10b encoding; two SKP Ordered Sets every 375 blocks for 128b/130b encoding). The DUT can either add or subtract SKP symbols, but must only add SKPs adjacent to received SKPs.

Test Pattern

14. The PTC completes sending the test pattern and then sends 4 consecutive EIOS, then transitions to Loopback.Exit.

15. In Loopback.Exit the PTC transmits the proper number of EIOS (one EIOS for 2.5 GT/s, 8 EIOS for 5.0 GT/s and 8.0 GT/s) and then enters electrical idle for 2 ms. The PTC checks that the DUT sends the same number of EIOS back. The PTC checks that the DUT then enters electrical idle.

16. The PTC then goes to Detect.

3.8.2.2 DUT is an Add-in Card or an Upstream port of a Switch or Bridge

1. Perform steps given in Section A.2.2 to reach Recovery.RcvrLock.

2. The PTC transmits TS1s with link and lane numbers set during the Configuration states. The speed_change bit is set to 1. The data rates that are advertised are selected from the test case (starting at the first test case). After receiving 8 consecutive TS1s or 8 consecutive TS2s with identical link and lane numbers to the one being transmitted the PTC transitions to Recovery.RcvrCfg.

   TEST CASE 1: Advertised Data Rates = 2.5 GT/s

   TEST CASE 2: (For PCIe2.x or later testing) Advertised Data Rates = 2.5 GT/s and 5.0 GT/s

   TEST CASE 3: (For PCIe3.x or later testing) Advertised Data Rates = 2.5 GT/s and 5.0 GT/s and 8.0 GT/s

1. The PTC transmits TS2s with non-PAD link and lane numbers. If 8.0 GT/s is going to be the data rate of operation during loopback the PTC shall let the DUT start 8.0 GT/s using its own default Tx preset settings and not send any EQ TS1 or EQ TS2 preset requests. After receiving 8 consecutive TS2s with speed_change bit set to 1, and after transmitting 32 TS2s with speed_change bit set to 1 after receiving the first TS2 and without interruption by an EIEOS, the PTC transitions to Recovery.Speed. If the 32 TS2s are interrupted by an EIEOS, the count is reset and the 32 TS2s are retransmitted.

2. In Recovery.Speed the PTC transmits 1 EIOS and goes to electrical idle. The PTC changes to the highest common advertised data rate within 0.5 ms. The next state is Recovery.RcvrLock at the new data rate.

3. If the operating data rate is 8.0 GT/s then the PTC transitions to Recovery.Equalization from Recovery.RcvrLock and Link Equalization is performed. If the operating data rate is 2.5 GT/s or 5.0 GT/s then the PTC stays in Recovery.RcvrLock and skips to step 9.

4. The PTC enters Phase 1 of Recovery.Equalization state whereas the DUT enters Phase 0. The PTC transmits TS1s with EC = 01b, Tx equalization sets the presets of 0x0, and reflects its current FS, LF, and post-coefficient values. The PTC should first see the DUT sending TS1s with EC = 00b reflecting the preset value it has requested in the EQ TS2s. After receiving 2 consecutive TS1s with EC = 01b within 2 ms the PTC transitions to Phase 2. If the PTC does not receive 2 consecutive TS1s with EC = 01b within 24 ms, the DUT fails.

5. In Phase 2 the PTC transmits TS1s with EC = 10b, reflecting its current coefficient values. If the PTC receives 2 consecutive TS1s with EC = 10b and requesting a new preset or coefficient, it processes these normally by either accepting or rejecting it as required (a rejected preset or coefficient is not a failure). After receiving 2 consecutive TS1s with EC = 11b the PTC transitions to Phase 3. If the PTC does not receive 2 consecutive TS1s with EC = 11b within 32 ms, the DUT fails.

6. In Phase 3, the PTC transmits TS1s with EC=11b, with the Use Preset bit (symbol 6, bit 7) set to a 1, and transmitter preset values (symbol 6, bits 6:3) set to 0x0. The PTC must transmit these TS1 for at least 1 μs. The PTC shall obtain new block alignment every time it receives the new coefficients. The PTC should check that the expected coefficients should be reflected within (500 + roundtrip time + logic delay) ns in the received TS1 (symbol 6, bits 6:3) and it matches the expected value of 0x0, and also should verify that the Reject Coefficient Values bit (symbol 9, bit 6) matches the expected 0 value. If not, the DUT fails the test. The PTC transitions to Recovery.RcvrLock.

7. In Recovery.RcvrLock the PTC transmits TS1s with EC = 00b. The PTC sets speed_change bit to 0. After receiving 8 consecutive TS1s or 8 consecutive TS2s with identical link and lane numbers the PTC transitions to Recovery.RcvrCfg.

8. In Recovery.RcvrCfg the PTC transmits TS2s. After receiving 8 consecutive TS2s with speed_change bit set to 0 and at least 16 consecutive TS2s have been transmitted after receiving the first TS2, the PTC transitions to Loopback.Entry via Recovery.Idle (which is just a pass through state).

9. In Loopback.Entry the PTC transmits TS1s with Compliance Receive bit set to 0, Hot Reset bit set to 0, Disable Link bit set to 0, and Loopback bit set. When the PTC receives 2 consecutive TS1s with the Loopback bit set to 1 from the DUT, the DUT is in Loopback.Active. If it does not see 2 consecutive TS1s with the Loopback bit set to 1 from the DUT within 2 ms, the test fails.

10. The PTC starts sending the test pattern. The DUT must replicate the data it receives and transmits it back to the PTC. The PTC checks that the received data matches the transmitted data (except for SKPs). The PTC sends SKP ordered sets properly (one SKP Ordered Set every 1538 symbol times for 8b/10b encoding; two SKP Ordered Sets every 375 blocks for 128b/130b encoding). The DUT can either add or subtract SKP symbols, but must only add SKPs adjacent to received SKPs.

Test Pattern

11. The PTC completes sending the test pattern, then sends 4 consecutive EIOS, and then transitions to Loopback.Exit.

12. In Loopback.Exit the PTC transmits the proper number of EIOS (one EIOS for 2.5 GT/s, 8 EIOS for 5.0 GT/s and 8.0 GT/s) and then enters electrical idle for 2 ms. The PTC checks that the DUT sends the same number of EIOS back. The PTC checks that the DUT then enters electrical idle.

13. The PTC then goes to Detect.

3.9 Function Level Reset (FLR)

These tests check that the DUT operates correctly at the link level when function level resets are performed. The tests only apply to Endpoints and then only if the DUT supports FLR.

The following rules will be checked as a part of this test suite:

1. Behavior of a function during FLR:
   a. Send a completion for the Config Write that initiated the FLR, before initiating the FLR.
   b. Complete FLR within 100 ms.
   c. Send CRS Completion Status to any Config Request after 100 ms, if extra time is required for function specific resets.
2. FLR does not affect the DUT's physical and data link layer states (i.e., negotiated link width and speed).

3.9.1 Test 61-10 Check the Behavior of the DUT During FLR

This test checks whether the DUT correctly performs the FLR. It checks the DUT's behavior during the FLR. Prerequisite to the test is that the function exists (i.e., it does not return Unsupported Request when reading Config Space register 00h) and that the function returns a 1 in the Function Level Reset Capability bit in the Device Capabilities register.

1. Perform the steps in Section A.2.2 or Section A.2.3 or Section A.2.4 to reach L0 at the desired speed.

2. The PTC shall set Bus Master Enable, Memory Space Enable, IO Space Enable, and Interrupt Disable bits in the Command register all to 1. After these bits are written the PTC shall issue a Configuration Read request to read the same bits back. If none of these bits are set to 1, then the FLR tests are still carried out, but no mechanism to check for successful FLR completion will exist. If at least one of these bits is set, the PTC shall enforce the register value checking in steps 5 and 6.

3. The PTC shall issue a Configuration Write Request to set the Initiate Function Level Reset bit to 1 in the Device Control register (offset 08h).

4. The PTC shall keep monitoring for a Configuration Write Completion for the request it had sent. If the PTC does not receive the Configuration Write Completion with Successful Completion status from the DUT within the 100 ms, the DUT fails the test. If the PTC receives the Configuration Write Completion, it shall move to the next step.

5. The PTC shall start a 100 ms timer. After the end of the 100 ms the PTC shall issue a Configuration Read Request to read the Command register (offset 04h) in the DUT. If the PTC receives a Configuration Read Completion with any completion status other than Successful Completion or CRS, the DUT fails the test. If the PTC does not receive any Configuration Read Completion, the DUT fails the test.

6. Some devices might take more than 100 ms to reinitialize following a FLR. In such cases the PTC may receive a Configuration Request Retry Completion Status for the Configuration Read Request issued in step 5. In such cases the PTC shall issue a new Configuration Read Request to read the Command register (offset 04h) in the DUT again, after every Configuration Read Completion with CRS status that it receives. If the PTC does not receive a Configuration Read Completion with Successful Completion status after 1 s the DUT fails the test, otherwise go onto the next step.

7. Using the Configuration Read Completion with Successful Completion status, the Bus Master Enable, Memory Space Enable, IO Space Enable, and Interrupt Disable bits of the Command register are checked. If all of these bits that returned a 1 in step 2 are now reset to 0, then the DUT passes the test.

8. Repeat the above steps for all supported data rates.

3.9.2 Test 61-20 Check Whether the Physical and Data Link Layers are Reset After an FLR

This test checks that the FLR does not impact the DUT's link states. Prerequisite to the test is that the function exists (i.e., it does not return Unsupported Request when reading Config Space register 00h) and that the function returns a 1 in the Function Level Reset Capability bit in the Device Capabilities register.

1. Perform the steps in Section A.2.2 or Section A.2.3 or Section A.2.4 to reach L0 at the desired speed.

2. The PTC shall perform a Configuration Read Request to read the DUT's Link Status register and store the value for comparison later.

3. After receiving a Configuration Read Completion the PTC shall perform a Configuration Write Request to set the Initiate Function Level Reset bit to 1 in the Device Control register (offset 08h) to initiate the FLR.

4. The PTC shall wait for the Configuration Write Completion from the DUT. The PTC shall monitor its link state and check that it does not leave L0. If it leaves L0s the test fails. If the

PTC does not receive the Configuration Write Completion with Successful Completion status from the DUT within the 100 ms, the DUT fails the test. If the PTC receives the Configuration Write Completion, it goes to the next step.

5. The PTC shall start a 100 ms timer. After the end of the 100 ms the PTC shall issue a Configuration Read Request to read the DUT's Link Status register. If the PTC receives a Configuration Read Completion with any completion status other than Successful Completion or CRS, the DUT fails the test. If the PTC does not receive any Configuration Read Completion the DUT fails the test.

6. Some devices might take more than 100 ms to reinitialize following a FLR. In such cases the PTC may receive a Configuration Request Retry Completion Status for the Configuration Read Request issued in step 5. In such cases the PTC shall issue a new Configuration Read Request to read the Command register (offset 04h) in the DUT again, after every Configuration Read Completion with CRS status that it receives. If the PTC does not receive a Configuration Read Completion with Successful Completion status after 1 sec the DUT fails the test, otherwise go onto the next step.

7. Using the Configuration Read Completion with Successful Completion status, if the Current Link Speed (bits [3:0]) and Negotiated Link Width (bits [9:4]) field values read before and after the FLR match, the DUT passes the test. If the values do not match, the DUT fails the test.

8. Repeat the above steps at all supported data rates.

3.10 Latency Tolerance Requests (LTR)

These tests check that the DUT sends properly formed LTR requests in several cases where responses are required. All these tests are applicable only for devices which support LTR capability, by checking the LTR Mechanism Supported bit in the Device Capabilities 2 register. If this bit returns 1, then the DUT's Port supports LTR and the test proceeds. Following rules are to be checked:

1. The DUT shall transmit a LTR message when the LTR Mechanism Enable bit is set.

2. If the LTR Mechanism Enable bit is cleared, and if the DUT has sent an LTR Message with Requirement bits set, then the Device must send a new LTR Message with those bits cleared.

3. When a DUT is directed to a non-D0 state by writing to its PMCSR, and if the DUT has sent an LTR Message with Requirement bits set, then the Device must send a new LTR Message with those bits cleared.

4. Multi-Function Device must transmit a conglomerated LTR Message which reflects the lowest value associated with any Function.

3.10.1 Test 62-10 Check if Upstream Port DUT Sends LTR Message After the LTR Enable has been Set/Cleared and the Format of the Message

This test checks to see if the DUT correctly transmits an LTR Message after the LTR Mechanism Enable bit has been cleared.

1. Perform the steps in Section A.2.2 or Section A.2.3 or Section A.2.4 to reach L0 at the appropriate speed.
2. During this test, the PTC issues a number of Configuration Write Requests. For each request issued the PTC shall wait for the Configuration Write Completion from the DUT. After receiving it, the PTC shall send an acknowledgement TLP. The PTC shall also acknowledge any other TLPs that the DUT had sent.
3. The PTC shall issue a Configuration Write Request to set the Max Snoop Latency and Max No-Snoop Latency registers to the value of 0C01h (corresponding to 32,768 ns).
4. The PTC shall issue a Configuration Write Request to set the LTR Mechanism Enable bit in Function 0 of the DUT (bit 10 in the Device Control 2 register).
5. The PTC shall keep monitoring for an LTR Message from the DUT. If the PTC receives an LTR Message where at least one Requirement bit is Set (bit 15 of either the Snoop Latency or No-Snoop Latency fields), or if the PTC has been in this step for at least 1 s then go to the next step.
6. The PTC shall issue a Configuration Write Request to clear the LTR Mechanism Enable bit.
7. The PTC shall keep monitoring for an LTR message from the DUT. If the PTC has been in this step for at least 1 s then go to the next step.
8. For any LTR Message received in this test, the PTC shall check whether the Fmt field (bits [7:5] of the first byte of the message received) is 001b, the Type is 10100b (Bits [4:0] of the first byte), the TC field is 000b (bits [6:4] of the second byte), and the Message Code is 00010000b (eighth byte).
9. For any LTR Message received in this test, if a Requirement bit is set in either the Snoop Latency or No Snoop Latency field, then the PTC shall check whether the LatencyValue (bits [9:0]) and LatencyScale (bits [12:10]) fields associated with that Requirement bit are valid. (Values in these fields must not use reserved encodings and must be less than or equal to the corresponding Max Snoop Latency or Max No-Snoop Latency value that was written above.)
10. For any LTR Message received in this test, if a Requirement bit is Set in that LTR Message, then the PTC shall check whether that that Requirement bit is Clear in a subsequent LTR Message.
11. If any check in steps 7 through 9 fails, the DUT fails the test. If all checks in steps 7 through 9 pass, and at least one LTR Message was received where at least one Requirement bit was Set, the DUT passes the test. Otherwise, the test is skipped (i.e., no LTR Messages were received or all LTR Messages had both Requirement bits Clear).

12. Repeat the above steps at all supported data rates.

3.10.2 Test 62-20 Check if Upstream Port DUT Sends LTR Message After the Device has been Directed to Non-D0 State

This test checks whether the DUT sends a LTR message after it has been directed to a non-D0 active state by writing to its PMCSR.

1. Perform the steps in Section A.2.2 or Section A.2.3 or Section A.2.4 to reach L0 at the desired speed.
2. During this test, the PTC issues a number of Configuration Write Requests. For each request issued the PTC shall wait for the Configuration Write Completion from the DUT. After receiving it, the PTC shall send an acknowledgement TLP. The PTC shall also acknowledge any other TLPs that the DUT had sent.
3. The PTC shall note if the DUT is a multi-Function Device.
4. The PTC shall issue a Configuration Write Request to set the Bus Master Enable bit in the Command register. For a multi-Function Device, this Configuration Write is repeated for each implemented Function in the Device.
5. The PTC shall issue a Configuration Write Request to set the Max Snoop Latency and Max No-Snoop Latency registers to the value of 0C01h (corresponding to 32,768 ns).
6. The PTC shall issue a Configuration Write Request to set the LTR Mechanism Enable bit in Function 0 of the DUT (bit 10 in the Device Control 2 register).
7. The PTC shall keep monitoring for an LTR Message from the DUT. If the PTC has been in this step for at least 1 s then go to the next step.
8. The PTC shall keep monitoring for an LTR Message from the DUT. The PTC now sends a Configuration Write Request to the DUT's PMCSR to change the state to the D3 state (i.e., write 11b to bits [1:0] of the PMCSR). For a multi-Function Device, this Configuration Write is repeated for each implemented Function in the Device.
9. The PTC shall keep monitoring for an LTR Message from the DUT. If the PTC has been in this step for at least 1 s, then go to the next step.
10. The PTC sends a Configuration Write Request to the DUT's PMCSR to change the state back to D0 state (i.e., write 00b to bits [1:0] of the PMCSR). For a multi-Function Device, this Configuration Write is repeated for each implemented Function in the Device. During this step, the PTC shall ignore any LTR Messages received from the DUT.
11. The PTC shall issue a Configuration Write Request to clear the LTR Mechanism Enable bit of Function 0 of the DUT (bit 10 in the Device Control 2 register). During this step, the PTC shall ignore any LTR Messages received from the DUT.

12. For any LTR Message received in this test, the PTC shall check whether the Fmt field (bits [7:5] of the first byte of the message received) is 001b, the Type is 10100b (Bits [4:0] of the first byte), the TC field is 000b (bits [6:4] of the second byte), and the Message Code is 00010000b (eighth byte).

13. For any LTR Message received in this test, if a Requirement bit is Set in either the Snoop Latency or No-Snoop Latency field, then the PTC shall check whether the LatencyValue (bits [9:0]) and LatencyScale (bits [12:10]) fields associated with that Requirement bit are valid. (Values in these fields must not use reserved encodings and must be less than or equal to the corresponding Max Snoop Latency or Max No-Snoop Latency value that was written above.)

14. For any LTR Message received in this test, if a Requirement bit is Set in that LTR Message, then the PTC shall check whether that that Requirement bit is Clear in a subsequent LTR Message.

15. If any check in steps 10 through 12 fails, the DUT fails the test. If all checks in steps 10 through 12 pass, and at least one LTR Message was received where at least one Requirement bit was Set, the DUT passes the test. Otherwise, the test is skipped (i.e. no LTR Messages were received or all LTR Messages had both Requirement bits Clear).

16. The PTC shall end the test.

17. Repeat the above steps for all supported data rates.

3.10.3 Test 62-30 Check if Upstream Port DUT has LTR Message Sent by a Multifunction Device

This test is no longer part of this test specification.

3.11 Link Partner Enters and Exits Compliance Mode

3.11.1 Test 63-10 Force the DUT into Compliance and Then Train it Out (Informational)

This test checks that the DUT is able to function correctly after the link partner initially incorrectly enters compliance mode and then attempts to retrain/reset the link. If a device is presented with a passive Z_{RX-DC} termination, it thinks that there is a device at the other end and starts link training. It goes from Detect.Quiet to Detect.Active and onto Polling.Active. In Polling.Active, if it does not see any TS1s or TS2s from the other side and it times out the 24 ms timer and enters Polling.Compliance. This test aims at testing the exit behavior after a device has entered the compliance mode due to the above reasons, but then returns back to the normal link training sequence.

1. The PTC starts in Detect.Quiet and waits for 12 ms and then transitions to Detect.Active.

2. In Detect.Active the PTC performs receiver detection, and on detecting a receiver the PTC shall go to Electrical Idle on all its transmitter lanes. If it does not detect a receiver within 1 s, the test fails. The PTC shall make sure that it terminates all its RX lanes with 50Ω nominal resistance.

3. The PTC shall stay in electrical idle for 30 ms. This will ensure that the DUT will timeout the 24 ms timer in Polling.Active forcing it to transition to Polling.Compliance.

4. The PTC shall monitor the incoming data from the DUT. If the PTC starts seeing compliance pattern at 2.5 GT/s and -3.5 dB De-emphasis, it is an indication that the DUT has entered Polling.Compliance, (i.e., Compliance mode).

5. The PTC shall now enter Polling.Active. In Polling.Active state the PTC transmits TS1s with link and lane numbers set to PAD, Compliance Receive bit set to 0, Hot Reset bit set to 0, Disable Link bit set to 0, and Loopback bit set to 0. All data rates shall be advertised.

   TX_Data = (COM, PAD, PAD, N_FTS, 0x0E, 0x00, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A)

6. After seeing the TS1s transmitted by the PTC, the DUT should exit Polling.Compliance and enter Polling.Active at 2.5 GT/s and resume the training.

7. After transmitting 1024 TS1s if the PTC receives either a) 8 TS2s (or their complement) with link and lane numbers set to PAD or b) 8 TS1s (or their complement) with link and lane numbers set to PAD, the PTC transitions to Polling.Configuration. This proves that the DUT has successfully exited the compliance mode and resumed training. If not, the DUT fails the test.

   TX_Data = (COM, PAD, PAD, N_FTS, 0x0E, 0x00, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A)

8. The PTC shall end the test.

3.11.2 Test 63-20 The PTC Goes to Compliance and Lets the DUT Train the PTC Out of Compliance Mode (Informational)

In this test the PTC would go to Polling.Compliance and test whether the DUT can train it out of Polling.Compliance and back to Polling.Active. This is to simulate a device incorrectly entering compliance mode.

1. The PTC starts in Detect.Quiet and waits for 12 ms and then transitions to Detect.Active.

2. In Detect.Active PTC performs receiver detection, and on detecting a receiver the PTC shall go to electrical idle on all its transmitter lanes. If it does not detect a receiver within 1 s, the test fails. The PTC shall make sure that it terminates all its RX lanes with 50 Ω nominal resistance.

3. The PTC shall enter Polling.Compliance and start transmitting compliance pattern at 2.5 GT/s and -3.5 dB De-emphasis. This is to indicate to the DUT that PTC is in Polling.Compliance mode.

4. The PTC shall expect the DUT to now enter Polling.Active. In Polling.Active state the DUT transmits TS1s with link and lane numbers set to PAD, Compliance Receive bit set to 0, Hot Reset bit set to 0, Disable Link bit set to 0, and Loopback bit set to 0. The PTC shall keep monitoring for TS1. If the PTC does not receive any TS1s after X ms the DUT fails the test.
5. After seeing the TS1s transmitted by the DUT, the PTC shall exit Polling.Compliance and enter Polling.Active at 2.5 GT/s and resume the training.
6. After transmitting 1024 TS1s if the PTC receives either a) 8 TS2s (or their complement) with link and lane numbers set to PAD or b) 8 TS1s (or their complement) with link and lane numbers set to PAD, the PTC transitions to Polling.Configuration. This proves that the DUT has successfully trained the PTC out of compliance mode and resumed training. If not, the DUT fails the test.

   TX_Data = (COM, PAD, PAD, N_FTS, 0x0E, 0x00, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A)
7. The PTC shall end the test.

3.12 SKP Processing (Informative)

These tests check that the DUT is able to correctly process SKPs in a variety of legal formats. The test will check various SKP requirements in different states of the LTSSM, starting with training, L0 and Loopback.

The following rules will be checked for compliance:

1. For 8b/10b encoding a SKP shall be sent every 1180 to 1538 symbols.
2. In loopback, the loopback slave can either add or subtract SKP symbols from the SKP OS transmitted by the loopback master.
3. Electrical Idle resets all the counters and timers for the SKP OS processing.
4. 1 SKP OS to be transmitted after the last FTS in 8b/10b encoding. (Can we really test rules 4 and 5, i.e., can we force the DUT TX into L0s? We can take the RX into L0s.) (Check on ASPM, link in idle long enough to force into L0s.)
5. No SKP OS to be transmitted before transmitting N_FTS FTSs.
6. SKP OS to be transmitted within idle symbol times.
7. For 128b/130b encoding a SKP shall be sent every 370 to 375 blocks (i.e., every 5920 to 6000 symbols).
   a. Values required in the last 4 fields of the SKP OS as described in the test description.
8. In L0 SKP OS shall be preceded by a data block with an EDS packet when at 8.0 GT/s.
   a. Values required in the last 4 fields of the SKP OS as described in the test description.

9. When transmitting Modified Compliance Pattern at 128b/130b encoding, the Error_Status field is to be transmitted in the SKP OS.

    a. Values required in the last 4 fields of the SKP OS as described in the test description.

10. For a multilane link:

    a. Same length SKP OS should be transmitted simultaneously on all the lanes.

    b. When transmitting Modified Compliance Pattern at 128b/130b encoding, the Error_Status field is to be transmitted in the SKP OS on a per lane basis.

    c. Values required in the last 4 fields of the SKP OS as described in the test description.

3.12.1 Test 64-10 Loopback Behavior

In this test we train the DUT into Loopback through the Configuration.Linkwidth.Start substate. This path can be used to test the first 3 rules mentioned previously. This test is only performed with a link width of x1 on lane 0.

1. The PTC starts in Detect.Quiet and waits for 12 ms and then transitions to Detect.Active.

2. In Detect.Active PTC performs receiver detection, on detecting a receiver it transitions to Polling.Active, otherwise it shall go back to Detect.Quiet.

3. In Polling.Active the PTC transmits TS1s with link and lane numbers set to PAD, Compliance Receive bit set to 0, Hot Reset bit set to 0, Disable Link bit set to 0, and Loopback bit set to 1. Only the highest speed that is to be used in Loopback shall be advertised in addition to 2.5 GT/s.

4. The PTC shall transmit TS1. After transmitting TS1s for 2 ms the PTC transitions to Polling.Configuration.

    a. The PTC shall start a counter to count the number of TSs being transmitted. The counter starts with the transmission of the first symbol of the first TS being transmitted. The PTC shall transmit SKP OS every 1180 symbols.

    b. The PTC shall keep monitoring for SKP OS. It shall expect a SKP OS after receiving 73 TSs (1180 symbols) and before receiving 96 TSs (1538 symbols). If the PTC does not receive any TSs within this time window, the DUT fails the test; else, the PTC shall reset the TS counter every time it receives the SKP OS and proceed to the next step.

5. In Polling.Configuration the PTC transmits TS2s with Hot Reset bit set to 0, Disable Link bit set to 0, and Loopback bit set to 1 and link and lane numbers set to PAD. (write something to indicate that the SKP rules are still followed and that counter is not reset)

6. The PTC transitions to Configuration.Linkwidth.Start TS2s after 10 ms.

7. From Configuration.Linkwidth.Start the PTC enters Loopback.Entry and transmits TS1s (Configuration.Linkwidth.Start is just a bypass state). PTC is the loopback master and the DUT is the loopback slave.

8. If the current speed of operation is not the highest common speed then PTC shall transmit 16 TS1s with Compliance Receive bit set to 0, Hot Reset bit set to 0, Disable Link bit set to 0, Loopback bit set to 1, and Selectable De-emphasis bit set to the desired value.

9. The PTC shall transmit an EIOS and go to electrical idle for 1 ms. The PTC shall change the speed of operation to the highest common speed during this 1 ms.

    a. The PTC shall reset its counters for tracking the SKP OS.

10. After coming out of electrical idle, PTC transmits TS1s with Compliance Receive bit set to 0, Hot Reset bit set to 0, Disable Link bit set to 0, and Loopback bit set, at the new data rate. For 2.5 GT/s and 5.0 GT/s, 8b/10b encoding is used and for 8.0 GT/s 128/130 encoding is used. If 8.0 GT/s is going to be the speed of operation during Loopback, the PTC shall transmit the desired DUT Transmitter Preset using the EQ TS1.

    a. After coming out of electrical idle, the PTC shall be tolerant to any SKP OS that might come any time before 1538 symbols from the last SKP OS. This is because the transmitters are not required, but only encouraged to reset their SKP processing counters during electrical idle.

11. When the PTC sees identical TS1s being transmitted by the DUT, the DUT is in Loopback.Active.

12. The PTC starts modified compliance pattern. The PTC sends SKP ordered sets every 1180 bytes of data if at 2.5 or 5.0 GT/s or after every 370 blocks if at 8.0 GT/s. The PTC shall monitor the incoming data and make sure it is the same as was transmitted with the exception for SKP OS. The DUT can either add or subtract SKP symbols. If the received SKP OS are not of the below mentioned lengths, the DUT fails the test:

    a. In the case of 8b/10b encoding, the received SKP OS must have 2, 3, or 4 SKP symbols (K28.0).

    b. In the case of 128b/130b encoding, the received SKP OS must be 8, 12, 16, 20, or 24 symbols (AAh) long.

13. The above test is to be repeated for the PTC SKP scheduling interval of 1180 byte, 1359 bytes, and 1538 bytes respectively.

14. The above test is to be repeated for all supported data rates.

3.12.2 Test 64-20 L0 for 2.5 GT/s and 5.0 GT/s

In this test we train the DUT into L0. This test checks how the SKPs are processed during normal operation.

1. Perform the steps given in Section A.2.2 to reach L0 at 2.5 GT/s. The PTC shall set 255 as the N_FTS value in the TSx ordered sets. PTC shall note the N_FTS value advertised by the DUT. The PTC shall keep transmitting SKP OS every 1180 symbols being transmitted.

    a. If performing test for 5.0 GT/s, follow the steps in Section A.2.3 to reach L0 at 5.0 GT/s.

2. The PTC shall transition to L0. The PTC shall now keep transmitting logical idle symbols 0x00 as per the framing rules required in L0.

3. The PTC shall keep monitoring for SKP OS. If it does not receive SKP OS within 1180 and 1538 symbols from the last received SKP OS, the DUT fails the test. After the PTC receives the SKP OS during logical idle, the test is complete.

4. Repeat the above test for PTC SKP scheduling intervals of 1180, 1359 and 1538 symbols.

5. Repeat the above test for both 2.5 GT/s and 5.0 GT/s.

3.12.3 Test 64-30 L0s Behavior for 2.5 GT/s and 5.0 GT/s

In this test we train the DUT into L0 substate, do a Config Write to the DUTs Link Control register, and then enter L0s.

1. Perform steps given in Section A.2.2 to reach L0 at 2.5 GT/s. The PTC shall set 255 as the N_FTS value in the TSx ordered sets. PTC shall note the N_FTS value advertised by the DUT.

    a. If performing test for 5.0 GT/s, follow the steps in Section A.2.3 to reach L0 at 5.0 GT/s

2. The PTC shall not do a Config Write Request to the DUT's Link Control register to set the Extended Synch bit. After the PTC receives the Config Write Completion from the DUT it shall go to step 3.

3. The PTC shall send a single EIOS in order to make the DUT receiver to transition to L0s. The PTC transmitter shall transition to Tx_L0s.Entry.

4. The PTC shall transmit 1 EIOS if the test speed is 2.5 GT/s or 2 EIOS if the test speed is 5.0 GT/s and go to electrical idle. It shall transition to Tx_L0s.FTS after $T_{TX\text{-}IDLE\text{-}MIN}$ timeout (20 ns).

5. The PTC shall transmit 1 FTS if at 2.5 GT/s or four EIE symbols if at 5.0 GT/s before transmitting N_FTS FTSs advertised by the DUT.

6. The PTC shall keep monitoring for SKP OS. If it received SKP OS before receiving 255 FTSs, the DUT fails the tests.

7. The PTC shall transmit a single SKP OS after the N_FTS FTSs plus, any additional SKP OS due as per the clock compensation requirements.

8. The PTC shall monitor for a single SKP OS plus additional due SKP OS from the DUT after the PTC has received the N_FTS FTSs. If 1538 symbols have been received since the last SKP OS, and if the PTC does not receive 2 SKP OS at the end of N_FTS FTSs, then the DUT fails

the test. If less than 1180 symbols have been received since the last SKP OS and PTC receives 2 SKP OSs after the N_FTS FTSs, then the DUT fails the test.

9. In the case where the Extended Sync bit is set, the DUT might transmit 4096 FTSs. In this case, the PTC shall check for SKP OS as per the normal clock compensation requirements. Still there should not be any SKP OS interrupting the first N_FTS FTSs. (Split into two tests, with and without the extended synch bit.)
10. Repeat the above test for PTC SKP scheduling intervals of 1180, 1359, and 1538 symbols.
11. Repeat the above test with and without setting the Extended Synch bit.
12. Repeat the above test for both 2.5 GT/s and 5.0 GT/s.

3.12.4 Test 64-40 L0 Behavior for 8.0 GT/s

This test checks rule number 8. This test is only applicable at 8.0 GT/s.

3.12.4.1 DUT is a Motherboard or a Downstream Port of a Switch

1. Perform steps given in Section A.2.2.1 to reach Recovery at 2.5 GT/s.
2. Perform the steps given in Section A.2.4.1 to reach L0 at 8.0 GT/s.
3. The PTC shall check to see if it receives and EDS token before it receives the first SKP OS. If more than one SKP OS are expected, the PTC shall check to see if it receives a data block with EDS before each SKP OS. If not, the DUT fails the test. If the received SKP OS does not match the description given below the DUT fails the test. The LFSR value received should match the LFSR value of the PTC's receiver. If not, the DUT fails the test. The Data Parity bit received in the SKP OS should also match the Data Parity bit calculated by the PTC from the incoming data. If not, the DUT fails the test.

    a. The received SKP should match the following

        i. Symbols 0 through (4*N – 1) = AAh; N = 1, 2, 3, 4, 5

        ii. Symbol 4*N = E1h (SKP_END) symbol.

        iii. 4*N + 1= If prior block was data block
            Bit[7] = Data Parity
            Bit[6:0] = LFSR[22:16]
        Else
            Bit[7] = ~LFSR[22]
            Bit[6:0] = LFSR[22:16]

        iv. 4*N + 2 = LFSR[15:8]

        v. 4*N + 3 = LFSR[7:0]

4. The PTC shall keep transmitting idle symbols. The PTC shall keep transmitting SKP OS every 370 to 375 block, incrementing the interval by 1 block every time the SKP is due. Once the

interval is 375, the next SKP OS shall be due after 370 blocks. The PTC shall maintain an even data parity bit of all payload of all the data blocks it has transmitted since the last SDS or SKP OS, whichever was the latest. The transmitted SKP OS shall be (fixed interval for entire test, or variable interval in the same test?).

a. The transmitted SKP shall match the following

i. Symbols 0 through (4*N – 1) = AAh; N = 1, 2, 3, 4, 5

ii. Symbol 4*N = E1h (SKP_END) symbol.

iii. 4*N + 1= If prior block was data block
Bit[7] = Data Parity
Else
Bit[7] = ~LFSR[22]
Bit[6:0] = LFSR[22:16]

iv. 4*N + 2 = LFSR[15:8]

v. 4*N + 3 = LFSR[7:0]

5. The PTC shall keep monitoring for SKP OS every 370 to 375 blocks from the last SKP OS received. If not, the DUT fails the test. If the received SKP OS does not match the description given below, the DUT fails the test.

6. The PTC shall end the test.

3.12.4.2 DUT is an Add-in Card or an Upstream Port of a Switch

1. Perform steps given in Section A.2.2.1 to reach Recovery at 2.5 GT/s.

2. Perform the steps given in Section A.2.4.2 to reach L0 at 8.0 GT/s.

3. The PTC shall check to see if it receives an EDS token immediately before it receives the first SKP OS. If more than one SKP OS are expected, the PTC shall check to see if it receives a data block with EDS immediately before each SKP OS. If not, the DUT fails the test. If the received SKP OS does not match the description given below the DUT fails the test. The LFSR value received should match the LFSR value of the PTC's receiver. If not, the DUT fails the test. The Data Parity bit received in the SKP OS should also match the Data Parity bit calculated by the PTC from the incoming data. If not, the DUT fails the test.

a. The received SKP should match the following:

i. Symbols 0 through (4*N – 1) = AAh; N = 1, 2, 3, 4, 5

ii. Symbol 4*N = E1h (SKP_END) symbol.

iii. 4*N + 1= If prior block was data block
Bit[7] = Data Parity
Bit[6:0] = LFSR[22:16]
Else

Bit[7] = ~LFSR[22]
Bit[6:0] = LFSR[22:16]

iv. 4*N + 2 = LFSR[15:8]

v. 4*N + 3 = LFSR[7:0]

4. The PTC shall keep transmitting idle data. The PTC shall keep transmitting SKP OS every 370 to 375 block, incrementing the interval by 1 block every time the SKP is due. The PTC shall maintain an even data parity bit of all payload of all the data blocks it has transmitted since the last SDS or SKP OS, whichever was the latest. The transmitted SKP OS shall be:

   a. The received SKP should match the following

      i. Symbols 0 through (4*N – 1) = AAh; N = 1, 2, 3, 4, 5

      ii. Symbol 4*N = E1h (SKP_END) symbol.

      iii. 4*N + 1= If prior block was data block
      Bit[7] = Data Parity
      Bit[6:0] = LFSR[22:16]
      Else
      Bit[7] = ~LFSR[22]
      Bit[6:0] = LFSR[22:16]

      iv. 4*N + 2 = LFSR[15:8]

      v. 4*N + 3 = LFSR[7:0]

5. The PTC shall keep monitoring for SKP OS every 370 to 375 blocks from the last SKP OS received. If not, the DUT fails the test. If the received SKP OS does not match the description given below, the DUT fails the test.

6. The PTC shall end the test.

3.12.5 Test 64-50 Modified Compliance Pattern at 8.0 GT/s Behavior

This test only applies to 8.0 GT/s. This test checks the behavior of SKP processing by the DUT when transmitting modified compliance pattern at 8.0 GT/s.

1. The PTC starts in Detect.Quiet and waits for 12 ms and then transitions to Detect.Active.

2. In Detect.Active PTC performs receiver detection, and on detecting a receiver it transitions to Polling.Active. On the absence of a receiver it goes back to the Detect.Quiet state.

3. In Polling.Active state the PTC transmits TS1s with link and lane numbers set to PAD, the Compliance Receive bit set to 0, Hot Reset bit set to 0, Disable Link bit set to 0, and Loopback bit set to 0. All data rates shall be advertised. The PTC shall keep transmitting SKP OS every 1359 symbols, with the count starting with the first symbol of the first TS1 being transmitted.

4. After transmitting 1024 TS1s and receiving either a) 8 TS2s (or their complement) with link and lane numbers set to PAD or b) 8 TS1s (or their complement) with link and lane numbers set to PAD, the PTC transitions to Polling.Configuration.

5. In Polling.Configuration the PTC transmits TS2s with Hot Reset bit set to 0, Disable Link bit set to 0, Loopback bit set to 0, and link and lane numbers set to PAD. The Compliance Receive bit (symbol 5 bit 4) shall be set to 1. The PTC shall transmit the Transmitter Preset it wants the DUT to use in compliance, in the symbol 6 (bits 6:3) of the TS1s. The PTC transitions to Polling.Compliance after 24 ms timeout.

6. The PTC shall send 1 EIOS and enter electrical idle. It shall switch to 8.0 GT/s within 1 ms. On coming out of electrical idle, it shall start transmitting Modified Compliance Pattern using the transmitter preset settings if they were requested by the DUT, else it can start with its own default preset.

7. The PTC shall keep monitoring for modified compliance pattern. Once it gets a pattern lock the PTC shall initialize an 8 bit Error_Status register to 00h. Bit number 8 is used as a pattern lock indicator. It shall be set once the PTC acquires pattern lock on the incoming modified compliance pattern. Anytime the there is a mismatch between incoming data payload in the Modified Compliance pattern which is supposed to be scrambled 00h, the Error_Status shall be increment by 1. The Error _Status saturates at 127. The PTC shall keep transmitting SKP OS every 370 to 375 blocks, incrementing the interval by one block every time the SKP is due.

    a. The transmitted SKP should match the following:

        i. Symbols 0 through (4*N – 1) = AAh; N = 1, 2, 3, 4, 5

        ii. Symbol 4*N = E1h (SKP_END) symbol

        iii. 4*N + 1= AAh

        iv. 4*N + 2 = Error_Status

        v. 4*N + 3 = ~Error_Status

8. The PTC shall keep monitoring for incoming SKP OS. If the PTC does not receive any SKP OS within 370 to 375 blocks from the previous SKP OS, the DUT fails the test.

    a. The received SKP should match the following:

        i. Symbols 0 through (4*N – 1) = AAh; N = 1, 2, 3, 4, 5

        ii. Symbol 4*N = E1h (SKP_END) symbol

        iii. 4*N + 1= AAh

        iv. 4*N + 2 = Error_Status

        v. 4*N + 3 = ~Error_Status

9. The PTC shall insert errors in the modified compliance pattern by transmitting a 01h instead of 00h. If the PTC does not see an increase in the Error_Status in the next SKP OS, indicating the

exact number of times the PTC transmitted 01h, the DUT fails the test. Else the DUT passes the test.

10. The PTC shall end the test.

3.12.6 Test 64-60 Multilane Test

These tests are only for multilane links. These tests shall be run at the maximum supported link width. This test is only supported on PTCs that support greater than x1.

1. Run test 3.11.2 for maximum supported link width. The PTC shall now check whether SKP OSs of same the length are transmitted on all the lanes simultaneously. If not, the DUT fails the test. Else it passes the test.

2. Run test 3.11.4 for maximum supported link width. The PTC shall now check whether SKP OSs of the same length are transmitted on all the lanes simultaneously. If not, the DUT fails the test. Else it passes the test.

3. Run test 3.11.5 for maximum supported link width. The PTC shall now check whether SKP OSs of same length are transmitted on all the lanes simultaneously. If not, the DUT fails the test. Else it passes the test. The PTC shall inject errors on different lanes at different time by transmitting 01h in modified compliance pattern. The PTC shall check whether Error_Status transmitted in the SKP OS are independent for every lane. If not, the DUT fails the test; else it passes the test.

3.13 Test 65-10 L1 for D3 State

This test verifies that the DUT correctly requests L1 entry when software sets the DUT state to D3. All PCI Express devices are required to support the device states as defined in the PCI Bus Power Management Interface Specification, namely D0 and D3 (while D1 and D2 are optional). Any non-D0 state corresponds to the link state L1. This test verifies whether after changing the device state of the DUT, the link transitions to L1 state.

3.13.1 DUT is a Motherboard or a Downstream Port of a Switch or Bridge

1. The PTC shall perform the steps in Section A.2.2 or Section A.2.3 or Section A.2.4 to reach L0 at the desired speed.

2. The PTC transmits PM_Enter_L1 DLLPs repeatedly with no more than the correct symbol times of idle between each PM_Enter_L1 DLLP (for 2.5 GT/s or 5.0 GT/s this is 4 symbol times; for 8.0 GT/s this is 16 symbol times). The PTC will continue to transmit SKP Ordered Sets at the appropriate intervals between the PM_Enter_L1 DLLPs. While in this step the PTC will process normally any TLPs that are received. The PTC waits in this state until it receives

one PM_Request_Ack DLLP. If it does not receive one within 1 s, the test case is skipped. Once the PTC receives one PM_Request_Ack DLLP, it goes to the next step.

3. The PTC checks that no TLPs are received. If it receives any TLP in this step, the test fails. The PTC waits in this state until it receives 'x' (defined in step 7) number of PM_Request_Ack DLLPs, and if it does not receive them within 50 ms the test fails. After receiving PM_DLLP_COUNT_VALUE number of PM_Request_Ack DLLPs, the PTC goes to the next step.

4. The PTC transmits the proper number of EIOS (one EIOS for 2.5 GT/s; two EIOS for 5.0 GT/s; one EIOS for 8.0 GT/s) and then the PTC transitions its lanes to Electrical Idle. It then goes to the next step.

5. The PTC waits in this state until it sees the proper number of EIOS (one EIOS for 2.5 GT/s; two EIOS for 5.0 GT/s; one EIOS for 8.0 GT/s), and if it does not see this within 50 ms, the test fails. After seeing the proper number of EIOS, the PTC goes to the next step.

6. The PTC waits in this state until it sees the link go to Electrical Idle, and if it does not see this within 50 ms the test fails. After seeing the link go to Electrical Idle, the test case ends. This completes the entry to L1 state.

7. The test is repeated with x = 1, 2, 4.

8. Steps 1 through 7 are repeated for all supported data rates.

3.13.2 DUT is an Add-in Card or an Upstream Port of a Switch or Bridge

1. The PTC shall perform the steps in Section A.2.2 or Section A.2.3 or Section A.2.4 to reach L0 at the desired speed.

2. The PTC now sends a Configuration Write Request TLP to the DUT Command register to set the Bus Master Enable bit (bit 2).

3. The PTC shall wait for the Configuration Write Completion from the DUT. After receiving it, the PTC shall send an acknowledgement TLP. The PTC shall also acknowledge any other TLPs that the DUT had sent.

4. The PTC now sends a Configuration Write Request TLP to the DUT's PMCSR to change the state to D3 (i.e., write 11b to bits [1:0] of the PMCSR).

5. The PTC shall wait for the Configuration Write Completion from the DUT. After receiving it, the PTC shall send an acknowledgement TLP. The PTC shall also acknowledge any other TLPs that the DUT had sent.

6. After the PTC has acknowledged all the TLPs from the DUT, it shall start a timer. If an initial PM_Enter_L1 DLLP is not received within 1 s, the test fails. The PTC shall transmit PM_Request_Ack DLLP repeatedly after receiving 'x' (defined in step 6) PM_Enter_L1 DLLP

from the DUT. After transmitting the first PM_Request_Ack DLLP, the PTC shall start a second timer.

7. After the PTC sees the receiver lanes transition to electrical idle, it shall stop transmission of the DLLPs and transition its transmitter lines into electrical idle. This completes the entry to L1 state. If the DUT transmitter lanes do not transition to electrical idle within 50 ms the DUT fails the test.

8. The test is repeated with x = 1, 2, 4.

9. Steps 1 through 6 are repeated for all supported data rates.

3.14 Test 66-10 Test ASPM-L1

Test Introduction

The intent of this test is to verify that the DUT that has a downstream port will properly negotiate an ASPM L1 entry request and will either reject the request or will accept it and transition to the L1 state. It also verifies that the DUT that has an upstream port that supports ASPM L1 will correctly request ASPM L1 entry when the DUT determines it wants to enter the ASPM L1 state. ASPM L1 support is optional in a DUT. If supported in a DUT with an upstream port, when all the functions in a device are enabled for ASPM L1, the device's upstream port link should transition to the L1 state if the link is idle.

Notes:

1. Test applies to all PCI Express device and port types that have a link

3.14.1 Root Port Test

Topology: Platform Test Topology, PTC in Platform Test mode

Section Notes:

1. DUT is the initiator of training and the PTC is the initiator of L1.

Initial Conditions:

1. Software determines if the DUT supports crosslinks, by checking the Crosslinks Supported bit in the Link Capabilities 2 register. If the bit returns 1, then the port unconditionally supports crosslinks. If the bit returns 0, then the Max Link Speeds field of the Link Capabilities register is checked, and if it returns a value of 0100b or greater, then the DUT does not support crosslinks. If none of these conditions are satisfied, then the DUT's support of crosslinks is unknown.

2. Software determines the DUT's support of ASPM L1 by reading the Active State Power Management (ASPM) Support field (Link Capabilities register). It sets the test flag as follows:

    a. If Active State Power Management (ASPM) Support is 00b or 01b, then set ASPM-L1_SUPPORTED_FLAG to 0.

b. If Active State Power Management (ASPM) Support is 10b or 11b, then set ASPM-L1_SUPPORTED_FLAG to 1.

3. Platform is power cycled and platform is up and running, with drivers for the test platform loaded and functioning.

4. The link is in Detect state.

5. PTC is disarmed and no trigger conditions set up.

Note: Once all criteria for the platform LTSSM to move to the next LTSSM state have been met (as specified in Chapter 4 of the PCI Express Base Specification), the platform must make the transition to the next state within 500 μs.

Procedure:

1. Perform the steps given in Section A.2.2 or Section A.2.3 or Section A.2.4 to go to the L0 state.

2. Program the following values to the indicated registers in the DUT:

TEST CASE 1: L1 Disabled

a. Set Active State Power Management (ASPM) Control field (Link Control register) to 00b.

b. Set RETRY_L1_ENTRY_COUNT = 3.

TEST CASE 2: (Only if ASPM-L1_SUPPORTED_FLAG=1) L1 Enabled

a. Set Active State Power Management (ASPM) Control field (Link Control register) to 10b.

b. Set RETRY_L1_ENTRY_COUNT = 2.

3. The PTC transmits PM_Active_State_Request_L1 DLLPs repeatedly with no more than the correct symbol times of idle between each PM_Active_State_Request_L1 DLLP (for 2.5 GT/s or 5.0 GT/s this is 4 symbol times; for 8.0 GT/s this is 16 symbol times). The PTC will continue to transmit SKP Ordered Sets at the appropriate intervals between the PM_Active_State_Request_L1 DLLPs. While in this step the PTC will process normally any TLPs that are received while also doing the following:

TEST CASE 1: L1 Disabled

a. The PTC waits in this state until it receives one PM_Active_State_Nak message TLP. If it does not receive one within 10 μs, the test case fails. Once the PTC receives one PM_Active_State_Nak message TLP, it sets the L1_REQUEST_REJECTED_FLAG to 1 and goes to the next step.

TEST CASE 2: (Only if ASPM-L1_SUPPORTED_FLAG=1) L1 Enabled

a. The PTC waits in this state until it receives either one PM_Request_Ack DLLP or one PM_Active_State_Nak message TLP. If it does not receive either one within 10 μs, the test case fails. If the PTC receives one PM_Request_Ack DLLP, it sets the L1_REQUEST_REJECTED_FLAG to 0 and goes to the next step. If the PTC receives one PM_Active_State_Nak message TLP, it sets the L1_REQUEST_REJECTED_FLAG to 1 and goes to the next step.

4. If the L1_REQUEST_REJECTED_FLAG is 0 this step is skipped and the PTC goes to the next step. If the L1_REQUEST_REJECTED_FLAG is 1, the PTC does not send any PM DLLPs, and waits in this state for 10 μs. While in this step the PTC will process normally any TLPs that are received. The PTC checks if it receives either any PM_Request_Ack DLLPs or any PM_Active_State_Nak message TLPs and if it receives either the test fails. After 10 μs the following is performed:

   a. Decrement RETRY_L1_ENTRY_COUNT by 1.

   b. If RETRY_L1_ENTRY_COUNT > 0, then go back and repeat steps 3-4.

   c. If RETRY_L1_ENTRY_COUNT = 0, then the test case passes and go to step 9.

5. The PTC continues to transmit PM_Active_State_Request_L1 DLLPs repeatedly with no more than the correct symbol times of idle between each PM_Active_State_Request_L1 DLLP (for 2.5 GT/s or 5.0 GT/s this is 4 symbol times; for 8.0 GT/s this is 16 symbol times). The PTC will continue to transmit SKP Ordered Sets at the appropriate intervals between the PM_Active_State_Request_L1 DLLPs. While in this step the PTC checks that no TLPs are received. If it receives any TLP in this step, the test fails. The PTC waits in this state until it receives PM_DLLP_COUNT_VALUE number of PM_Request_Ack DLLPs, and if it does not receive them within 50 ms the test fails. After receiving PM_DLLP_COUNT_VALUE number of PM_Request_Ack DLLPs, the PTC goes to the next step. (Talk about the value of PM_DLLP_COUNT_VALUE in the meeting.)

6. The PTC transmits the proper number of EIOS (one EIOS for 2.5 GT/s; two EIOS for 5.0 GT/s; one EIOS for 8.0 GT/s) and then the PTC transitions its lanes to Electrical Idle. It then goes to the next step.

7. The PTC waits in this state until it sees the proper number of EIOS (one EIOS for 2.5 GT/s; two EIOS for 5.0 GT/s; one EIOS for 8.0 GT/s), and if it does not see this within 50 ms, the test fails. After seeing the proper number of EIOS, the PTC goes to the next step.

8. The PTC waits in this state until it sees the link go to Electrical Idle, and if it does not see this within 50 ms, the test fails. After seeing the link go to Electrical Idle, the test case ends. Next the DUT is restored to its initial test conditions and the next test case is executed.

9. The entire test is repeated for each of the supported data rates.

10. If the DUT supports crosslinks, return the DUT to the initial conditions and then perform the steps listed in the procedure for Endpoint Device Test.

3.14.2 Endpoint Device Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. PTC is the initiator of training and the DUT is initiator of L1.

Initial Conditions:

1. Software determines if the DUT supports crosslinks, by checking the Crosslinks Supported bit in the Link Capabilities 2 register. If the bit returns 1, then the port unconditionally supports crosslinks. If the bit returns 0, then the Max Link Speeds field of the Link Capabilities register is checked, and if it returns a value of 0100b or greater, then the DUT does not support crosslinks. If none of these conditions are satisfied, then the DUT's support of crosslinks is unknown.

2. Software determines the DUT's support of ASPM L1 by reading the Active State Power Management (ASPM) Support field (Link Capabilities register). It sets the test flag as follows:

   a. If Active State Power Management (ASPM) Support is 00b or 01b, then set ASPM-L1_SUPPORTED_FLAG to 0.

   b. If Active State Power Management (ASPM) Support is 10b or 11b, then set ASPM-L1_SUPPORTED_FLAG to 1.

3. Platform is up and running, with drivers for the PTC loaded and functioning.

4. Fundamental Reset is asserted to the DUT.

5. The link is in Detect state.

6. PTC is disarmed and no trigger conditions set up.

Procedure:

1. Perform the steps given in Section A.2.2 or Section A.2.3 or Section A.2.4 to go to the L0 state.

2. Read the Power Management Capabilities register and set the following flags:

   a. If D1 Support is 0, then set D1_SUPPORTED_FLAG to 0.

   b. If D1 Support is 1, then set D1_SUPPORTED_FLAG to 1.

   c. If D2 Support is 0, then set D2_SUPPORTED_FLAG to 0.

   d. If D2 Support is 1, then set D2_SUPPORTED_FLAG to 1.

3. If the MULTIFUNCTION_DUT_FLAG is 1, then perform the following sequence starting at Function Number = 7 of the DUT and repeating the sequence for each lower Function Number until it stops after completing Function Number = 1 (i.e., FN = 7 to FN = 1).

   a. Read the Vendor ID register of Function Number = FN of the DUT, and if it returns a value of 0xFFFF or 0x0001, skip the remaining steps of this sequence and go onto the next Function Number.

   b. Program the test case values listed below to the indicated registers in Function Number = FN of the DUT.

   c. Read back the Power State field (Power Management Control/Status register) in Function Number = FN of the DUT until it returns EXPECTED_POWER_STATE_VALUE. If it does not return the correct value within 1 s, then the test case is skipped.

4. Program the following values to the indicated registers in Function Number = 0 of the DUT:

   TEST CASE 1: L1 Disabled, D0

   a. Set Active State Power Management (ASPM) Control field (Link Control register) to 00b.

   b. Set RETRY_L1_ENTRY_COUNT = 3.

   c. Set Power State (Power Management Control/Status register) to 00b.

   d. Set EXPECTED_POWER_STATE_VALUE = 00b.

   TEST CASE 2: (Only if ASPM-L1_SUPPORTED_FLAG=1) L1 Enabled, D0, L1 Rejected Once

   a. Set Active State Power Management (ASPM) Control field (Link Control register) to 10b.

   b. Set RETRY_L1_ENTRY_COUNT = 2.

   c. Set Power State (Power Management Control/Status register) to 00b.

   d. Set EXPECTED_POWER_STATE_VALUE = 00b.

   TEST CASE 3: Only if ASPM-L1_SUPPORTED_FLAG=1) L1 Enabled, D0, L1 Accepted

   a. Set Active State Power Management (ASPM) Control field (Link Control register) to 10b.

   b. Set RETRY_L1_ENTRY_COUNT = 2.

   c. Set Power State (Power Management Control/Status register) to 00b.

   d. Set EXPECTED_POWER_STATE_VALUE = 00b.

5. Read back the Power State field (Power Management Control/Status register) in Function Number = 0 of the DUT until it returns EXPECTED_POWER_STATE_VALUE. If it does not return the correct value within 1 s, then the test case is skipped.

6. The PTC checks for the following:

   TEST CASE 1: L1 Disabled

   a. No PM_Active_State_Request_L1 DLLPs are received. If it does not receive one within 1 s, the test case passes and goes to step 11.

   TEST CASE 2: (Only if ASPM-L1_SUPPORTED_FLAG=1) L1 Enabled, L1 Rejected Once

   a. At least one PM_Active_State_Request_L1 DLLPs is received. If it does not receive one within 1 s, the test case is skipped. Once the PTC receives one PM_Active_State_Request_L1 DLLP, it goes to the next step.

   TEST CASE 3: (Only if ASPM-L1_SUPPORTED_FLAG=1) L1 Enabled, L1 Accepted

   a. At least one PM_Active_State_Request_L1 DLLPs is received. If it does not receive one within 1 s, the test case is skipped. Once the PTC receives one PM_Active_State_Request_L1 DLLP, it goes to the next step.

7. The PTC checks that no TLPs are received. If it receives any TLP in this step, the test fails. The PTC waits in this state until it receives PM_DLLP_COUNT_VALUE number of PM_Active_State_Request_L1 DLLPs, and if it does not receive them within 50 ms the test fails. After receiving PM_DLLP_COUNT_VALUE number of PM_Active_State_Request_L1 DLLPs, the PTC goes to the next step. (Talk about the value of PM_DLLP_COUNT_VALUE in the meeting.)

8. For Test Case 3, skip to step 13. For Test Case 2, continue with steps 9-12.

9. The PTC transmits one PM_Active_State_Nak message TLP. While in this step the PTC checks that no TLPs are received. If it receives any TLPs in this step, the test fails. The PTC waits in this state until it receives an ACK DLLP, and if it does not see this within 50 ms, the test fails. After seeing the ACK DLLP, the PTC goes to the next step.

10. The PTC starts a timer that counts down for 9.5 μs. If the timer reaches zero, the PTC goes to the next step. Every time the PTC receives a PM_Active_State_Request_L1 DLLP the timer is reset to 9.5 μs and starts to count down again. If the timer has not expired within 50 ms, the test fails.

11. The PTC checks that at least one PM_Active_State_Request_L1 DLLPs is received. If it does not receive one within 1 s, the test case is skipped. Once the PTC receives one PM_Active_State_Request_L1 DLLP, it goes to the next step.

12. The PTC checks that no TLPs are received. If it receives any TLP in this step, the test fails. The PTC waits in this state until it receives PM_DLLP_COUNT_VALUE number of PM_Active_State_Request_L1 DLLPs, and if it does not receive them within 50 ms, the test fails. After receiving PM_DLLP_COUNT_VALUE number of PM_Active_State_Request_L1 DLLPs, the PTC goes to the next step. (Talk about the value of PM_DLLP_COUNT_VALUE in the meeting.)

13. The PTC transmits PM_Request_Ack DLLPs repeatedly with no more than the correct symbol times of idle between each PM_Request_Ack DLLP (for 2.5 GT/s or 5.0 GT/s this is 4 symbol times; for 8.0 GT/s this is 16 symbol times). The PTC will continue to transmit SKP Ordered Sets at the appropriate intervals between the PM_Request_Ack DLLPs. While in this step the PTC checks that no TLPs are received. If it receives any TLPs in this step, the test fails. The PTC waits in this state until it receives the proper number of EIOS (one EIOS for 2.5 GT/s; two EIOS for 5.0 GT/s; one EIOS for 8.0 GT/s), and if it does not see this within 50 ms, the test fails. After seeing the proper number of EIOS, the PTC goes to the next step.

14. The PTC transmits the proper number of EIOS (one EIOS for 2.5 GT/s; two EIOS for 5.0 GT/s; one EIOS for 8.0 GT/s). The PTC waits in this state until it sees the link go to Electrical Idle, and if it does not see this within 50 ms, the test fails. After seeing the link go to Electrical Idle, the PTC goes to the next step.

15. The PTC transitions its lanes to Electrical Idle and the test case ends. Next the DUT is restored to its initial test conditions and the next test case is executed.

16. The entire test is repeated for each of the supported data rates.

17. If the DUT supports crosslinks, return the DUT to the initial conditions and then perform the steps listed in the procedure for Root Port Test.

3.14.3 Switch and Bridge Downstream Port Test

Topology: Switch Test Topology, PTC in Platform Test mode

Section Notes:

1. Algorithm same as in the Root Port Test except the DUT is a Switch's or Bridge's downstream port.

3.14.4 Switch and Bridge Upstream Port Test

Topology: Endpoint Test Topology, PTC in Add-in card Test mode

Section Notes:

1. Algorithm same as in the Endpoint Device Test except the DUT is a Switch's or Bridge's upstream port.

A

Appendix A. MACROS

A.1 Protocol Test Card (PTC) Related

These Macros shall apply to any test equipment that has the test capabilities listed in Chapter 2. As noted earlier, in this document such capable test equipment is referred to as the PTC. The implementation details of the Macros are test equipment specific.

A.1.1 MACRO_POLL_PTC_FOR_LINK_RETRAINING ()

Description: Monitor the link status of the PTC after a link retrain. This macro will periodically check the link status for retraining to complete. If the link does not successfully retrain within the specified period, this will be treated as a test failure and the test is aborted.

A.1.2 MACRO_POLL_DUT_FOR_LINK_RETRAINING ()

Description: Monitor the link status of the Root Port after a link retrain. This macro will periodically check the link status for retraining to complete. If the link does not successfully retrain within the specified period, this will be treated as a test failure and the test is aborted.

A.1.3 MACRO_PTC_ARM ()

Description: Enables the PTC card to act on the test conditions already programmed into it. This along with the MACRO_PTC_DISARM capability gives the software control on when the PTC shall apply the test conditions on the DUT. This will also enable capturing the traffic flowing through the PTC into its trace buffer.

A.1.4 MACRO_PTC_DISARM ()

Description: Disables the PTC card from applying the test conditions to any traffic flowing through the PTC and this will also stop its trace buffer acquisition.

A.1.5 MACRO_PTC_STATUS (ACTION_COUNT)

Description: Determine how far the PTC is in executing the test command set up. This macro will periodically poll the action count for a maximum of 10 ms for the count to go to zero. If the count does not reach zero in that period, this will be treated as a test failure and the test is aborted.

ACTION_COUNT – 0 => PTC executed on the test command

!= 0 > PTC is still looking to assert the test command

A.1.6 MACRO_PTC_CONFIG_TRACE_BUF (FILTER, DIRECTION)

Description: Program the PTC to capture desired traffic.

FILTER – The amount of packet information which the PTC will capture in the supplied direction

TLP_HEADERS_ONLY – capture only TLP headers in the PTC buffer

TRAINING_SETS_ONLY – capture only TS1 or TS2 Ordered Sets in the PTC buffer

DIRECTION – A matching direction of traffic on which the PTC will capture

DOWNSTREAM_DIR – capture only downstream traffic in the PTC buffer

UPSTREAM_DIR – capture only upstream traffic in the PTC buffer

A.1.7 MACRO_PTC_PROGRAM (PTC_ACTION, PATTERN_TO_MATCH, ACTION COUNT)

Description: Program the PTC to generate desired test condition.

PTC_ACTION – Refers to one of the test conditions below that the PTC will generate:

CORRUPT_RESERVED_FIELDS_ACK_DLLP - PTC will use non-zero values in at least one reserved field of the generated ACK DLLP

DELAY_ACK_NAK_LEGAL – PTC will delay the ACK DLLP or NAK DLLP for a period less than the ACK_NAK LATENCY requirement

NAK – PTC will generate a NAK DLLP instead of an ACK DLLP

NO_ACK_NAK – PTC will not generate an ACK DLLP or a NAK DLLP

CORRUPT_ACK_CRC – PTC will generate an ACK DLLP with a bad CRC

DLLP_UNDEFINED_ENCODING – PTC will generate a DLLP with an undefined encoding instead of an ACK DLLP. (The DLLP Type field will contain a value that is defined as Reserved in

the Base Specification revision for the test level selected, and that is not defined in any additional specifications (e.g., MR-IOV specification) for the test level selected.)

ACK_DLLP_WRONG_SEQ_NUM – PTC will generate an ACK DLLP with the wrong Sequence Number

GENERATE_ECRC – PTC will generate a TLP with a TLP Digest

CORRUPT_LCRC – PTC will generate a TLP with a bad LCRC

CORRUPT_ECRC – PTC will generate a TLP with a TLP Digest that contains a bad ECRC

DUPLICATE_TLP – PTC will generate two identical TLPs

PATTERN_TO_MATCH – A matching condition (TLP or DLLP as appropriate) on which the PTC will generate the test condition.

ACTION COUNT – A non-zero value that specifies how many times the PTC will generate the test condition.

A.1.8 MACRO_READ_CONFIG_DATA_FROM_PTC (QUAL)

Description: Read configuration data from the PTC. This could be a byte, word, dword, or greater, depending upon the qualifier.

QUAL – VENDOR_DEV_ID – default – first DWORD

- PCI_COMPATIBLE- first 256 bytes or as many as specified
  - Any other specific fields in PCI compatible space
  - Extended Config space
  - Any specific fields in extended Config space

A.1.9 MACRO_READ_CONFIG_DATA_FROM_KEP (QUAL)

Description: Read configuration data from the PTC. This could be a byte, word, dword, or greater depending upon the qualifier.

QUAL – VENDOR_DEV_ID – default – first DWORD

- PCI_COMPATIBLE- first 256 bytes or as many as specified
  - Any other specific fields in PCI compatible space
  - Extended Config space
  - Any specific fields in extended Config space

A.1.10 MACRO_READ_CONFIG_DATA_FROM_DUT (QUAL)

Description: Read configuration data from the DUT. This could be a byte, word, dword, or greater depending upon the qualifier.

QUAL – VENDOR_DEV_ID – default – first DWORD

- PCI_COMPATIBLE- first 256 bytes or as many specified
  - Any other specific fields in PCI compatible space
  - Extended Config space
  - Any specific fields in extended Config space

A.1.11 MACRO_READ_DATA_FROM_PTC (START_ADDR, BYTE_COUNT, BAR_NUM)

Description: Read from the memory behind the PTC's BAR. By default, this will be the trace buffer memory. There is at least one BAR in the PTC.

START_ADDR – Start address at which the read begins

BYTE_COUNT – Number of bytes to read

BAR_NUM – When there is more than one BAR implemented by the test equipment, this will specify the BAR behind which the memory read shall take place.

A.1.12 MACRO_READ_DATA_FROM_KEP (START_ADDR, BYTE_COUNT, BAR_NUM)

Description: Read from the memory behind the KEP's BAR. There is at least one BAR in the KEP.

START_ADDR – Start address at which the read begins

BYTE_COUNT – Number of bytes to read

BAR_NUM – When there is more than one BAR implemented by the test equipment, this will specify the BAR behind which the memory read shall take place.

A.1.13 MACRO_WRITE_DATA_TO_PTC (START_ADDR, DATA_PATTERN, BYTE_COUNT, BAR_NUM)

Description: Write to the memory behind the PTC' BAR. There is at least one BAR in the PTC.

START_ADDR – Start address at which the write begins

DATA_PATTERN – A DWORD pattern that will be repeated up to the byte count. In the case of partial fills, the lower bytes of the pattern shall be used as needed.

BYTE_COUNT – Number of bytes to write

BAR_NUM – When there is more than one BAR implemented by the test equipment, this will specify the BAR behind which the memory write shall take place.

A.1.14 MACRO_WRITE_DATA_TO_KEP (START_ADDR, DATA_PATTERN, BYTE_COUNT, BAR_NUM)

Description: Write to the memory behind the KEP's BAR. There is at least one BAR present in the KEP.

START_ADDR – Start address at which the write begins

DATA_PATTERN – A DWORD pattern that will be repeated up to the byte count. In the case of partial fills, the lower bytes of the pattern shall be used as needed.

BYTE_COUNT – Number of bytes to write

BAR_NUM – When there is more than one BAR implemented by the test equipment, this will specify the BAR behind which the memory write shall take place.

A.1.15 MACRO_PTC_CLEANUP ()

Description: Disarm the PTC and do any other clean up needed on the PTC (test equipment).

A.2 GENERAL

A.2.1 MACRO_ENABLE_LINK (ACTION)

Description: Enable and disable the link at any of the downstream ports on the PTC device.

ACTION – YES – Enables Link

NO – Disables Link

A.2.2 Steps to Enter L0 (or Recovery.RcvrLock) Using a PTC

The following sections describe standard sequences to enter L0, or Recovery.RcvrLock when using a PTC to test a DUT. Each step is followed by a recommended value of the training set in terms of symbols.

Symbol	Description/Value (in Hex)
COM	BC for 8b/10b 1E for 128b/130b
PAD	F7
N_FTS	10
LINK_RCVD	8 bit received link number
LANE_RCVD	8 bit received lane number
TXPRERxd	4 bit received TX Preset Request

A.2.2.1 DUT is a Motherboard or a Downstream Switch Port (2.5 GT/s)

1. The PTC starts in the Detect.Quiet state and waits for 12 ms and then transitions to Detect.Active.

2. In Detect.Active PTC performs receiver detection, on detecting a receiver it transitions to Polling.Active. On absence of a receiver it goes back to the Detect.Quiet state.

3. In Polling.Active state the PTC transmits TS1s with link and lane numbers set to PAD, the enter compliance bit set to 0 and loopback bit set to 0. All three speeds: 2.5 GT/s, 5.0 GT/s, and 8.0 GT/s shall be advertised.

   TX_Data = (COM, PAD, PAD, N_FTS, 0x0E, 0x00, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A)

4. After transmitting 1024 TS1s and receiving either a) 8 TS2s (or complement) with link and lane number set to PAD or b) 8 TS1s (or complement) with link and lane numbers set to PAD, the PTC transitions to Polling.Configuration.

   TX_Data = (COM, PAD, PAD, N_FTS, 0x0E, 0x00, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A)

5. In Polling.Configuration the PTC transmits TS2s with loopback bit set to 0 and link and lane numbers set to PAD. The PTC transitions to Configuration.Linkwidth.Start after receiving 8 TS2s with link and lane numbers set to PAD and transmitting 16 TS2s after receiving the first TS2.

   TX_Data = (COM, PAD, PAD, N_FTS, 0x0E, 0x00, 0x45, 0x45, 0x45, 0x45, 0x45, 0x45, 0x45, 0x45, 0x45, 0x45)

6. The PTC transmits TS1s with link and lane numbers set to PAD. After receiving 2 TS1s with the same link number that is not PAD and lane number set to PAD, PTC transitions to Configuration.Linkwidth.Accept.

   TX_Data = (COM, PAD, PAD, N_FTS, 0x0E, 0x00, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A)

7. PTC transmits TS1s with the link number set to the received link number and lane number set to PAD. After receiving 2 identical TS1s, PTC transmits TS1s with the received link and lane numbers. The next state is Configuration.Lanenum.Wait.

   TX_Data = (COM, LINK_RCVD, PAD, N_FTS, 0x0E, 0x00, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A)

8. The PTC transmits TS1s with link and lane number identical to the ones received. The PTC transitions to Configuration.Lanenum.Accept after receiving 2 TS2s.

   TX_Data = (COM, LINK_RCVD, LANE_RCVD, N_FTS, 0x0E, 0x00, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A)

9. In the Configuration.Lanenum.Accept state the PTC transmits TS1s with link and lane number identical to the ones received. Next state is Configuration.Complete after receiving 2 TS2s.

   TX_Data = (COM, LINK_RCVD, LANE_RCVD, N_FTS, 0x0E, 0x00, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A)

10. In Configuration.Complete state the PTC transmits TS2s with the same link and lane numbers. After 8 consecutive TS2s are received with identical data rate identifiers (including identical Link Upconfigure Capability bit (Symbol 4, bit 6)), and link and lane numbers same as the previous non-PAD values, and 16 TS2s are transmitted after receiving the first TS2, the next state is Configuration.Idle.

    TX_Data = (COM, LINK_RCVD, LANE_RCVD, N_FTS, 0x0E, 0x00, 0x45, 0x45, 0x45, 0x45, 0x45, 0x45, 0x45, 0x45, 0x45, 0x45)

11. In Configuration.Idle the PTC transmits idle symbols. After receiving 8 consecutive idle symbol times and transmitting 16 idle symbols after receiving the first idle symbol, the PTC transitions to either L0 or Recovery.RcvrLock if it wants to bypass L0.

    TX_Data = 0x00

A.2.2.2 DUT is an Add-in Card or an Upstream Switch Port (2.5 GT/s)

1. The PTC starts in the Detect.Quiet state and waits either for 12 ms or for Electrical Idle to be broken on any Lane and then transitions to Detect.Active.

2. In Detect.Active PTC performs receiver detection, on detecting a receiver it transitions to Polling.Active, otherwise it goes back to Detect.Quiet.

3. In Polling.Active state the PTC transmits TS1s with link and lane numbers set to PAD, the enter compliance bit set to 0 and loopback bit set to 0. All three speeds: 2.5 GT/s, 5.0 GT/s, and 8.0 GT/s shall be advertised.

   TX_Data = (COM, PAD, PAD, N_FTS, 0x0E, 0x00, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A)

4. After transmitting 1024 TS1s and receiving either a) 8 TS2s (or complement) with link and lane number set to PAD or b) 8 TS1s (or complement) with link and lane numbers set to PAD the PTC transitions to Polling.Configuration.

   TX_Data = (COM, PAD, PAD, N_FTS, 0x0E, 0x00, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A)

5. In Polling.Configuration the PTC transmits TS2s with loopback bit set to 0 and link and lane numbers set to PAD. The PTC transitions to Configuration.Linkwidth.Start after receiving 8 TS2s with link and lane numbers set to PAD and transmitting 16 TS2s after receiving the first TS2.

   TX_Data = (COM, PAD, PAD, N_FTS, 0x0E, 0x00, 0x45, 0x45, 0x45, 0x45, 0x45, 0x45, 0x45, 0x45, 0x45, 0x45)

6. The PTC transmits TS1s with link number set to 1 and lane numbers set to PAD. After receiving 1 TS1 with link and lane number set to PAD and 2 TS1s with the non-PAD link number that is transmitted by the PTC and lane number set to PAD, PTC transitions to Configuration.Linkwidth.Accept.

   TX_Data = (COM, 0x01, PAD, N_FTS, 0x0E, 0x00, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A)

7. In this substate the PTC transmits TS1s with the link number set to 1 and lane number set to 0. The next state is Configuration.Lanenum.Wait.

   TX_Data = (COM, 0x01, 0x00, N_FTS, 0x0E, 0x00, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A)

8. The PTC transmits TS1s with link and lane number set to 1 and 0 respectively. After receiving 2 TS1s with link and lane number set to 1 and 0 the PTC transitions to Configuration.Lanenum.Accept.

   TX_Data = (COM, 0x01, 0x00, N_FTS, 0x0E, 0x00, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A)

9. The PTC transmits TS1s with link and lane number set to 1 and 0 respectively. After receiving 2 TS1s with link and lane number set to 1 and 0 the PTC transitions to Configuration.Complete.

   TX_Data = (COM, 0x01, 0x00, N_FTS, 0x0E, 0x00, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A, 0x4A)

10. In Configuration.Complete state the PTC transmits TS2s with the same link and lane numbers set to 1 and 0 respectively. After 8 consecutive TS2s are received with identical data rate identifiers (including identical Link Upconfigure Capability bit (Symbol 4, bit 6)), and link and lane numbers set to 1 and 0 respectively, and 16 TS2s are transmitted after receiving the first TS2, the next state is Configuration.Idle.

    TX_Data = (COM, 0x01, 0x00, N_FTS, 0x0E, 0x00, 0x45, 0x45, 0x45, 0x45, 0x45, 0x45, 0x45, 0x45, 0x45, 0x45)

11. In Configuration.Idle the PTC transmits idle symbols. After receiving 8 consecutive idle symbol times and transmitting 16 idle symbols after receiving the first idle symbol, the PTC transitions to either L0 or Recovery.RcvrLock if it wants to bypass L0.

    TX_Data = 0x00

A.2.3 Procedure to Reach L0 at 5.0 GT/s

Follow these steps to reach L0 at 5.0 GT/s:

1. Follow the steps in Section A.2.2.1 or Section A.2.2.2 to reach Recovery.RcvrLock at 2.5 GT/s.

2. The PTC shall transmit TS1 with the link and lane number that were finalized in the Configuration state. The speed_change bit shall be set to 1 and 5.0 GT/s speed shall be advertised.

3. After receiving 8 TS1s or TS2s with same link and lane numbers the PTC shall transition to Recovery.RcvrCfg.

4. The PTC shall transmit TS2s with link and lane number set to the ones finalized during the Configuration state, speed_change bit set to 1 and 5.0 GT/s being advertised. After receiving 8 TS2s with speed_change bit set to 1 and 5.0 GT/s advertised, and after 32 TS2s with speed_change bit set to 1 have been transmitted after receiving the first TS2 with speed_change bit set to 1.

5. The PTC shall send an EIOS and go to electrical idle. The PTC shall switch to 5.0 GT/s at 0.5 ms. After 0.5 ms it shall reenter Recovery.RcvrLock at 5.0 GT/s.

6. The PTC shall transmit TS1s at 5.0 GT/s with non-PAD link and lane numbers and speed_change bit set to 0. After receiving 8 TS1s or TS2s with non-PAD link and lane numbers and speed_change bit set to 0, the PTC shall transition to Recovery.RcvrCfg.

7. The PTC shall transmit TS2s with non-PAD link and lane numbers and speed_change bit set to 0. After receiving 8 TS2s with speed_change bit set to 0 and after at least 16 TS2s have been transmitted after receiving the first TS2, the PTC shall transition to Recovery.Idle.

8. The PTC shall transmit idle data. After receiving 8 symbol times of idle data and after at least 16 idle symbols have been transmitted after receiving the first idle symbol the PTC shall transition to L0.

A.2.4 Procedure to Reach L0 at 8.0 GT/s

A.2.4.1 DUT is a Motherboard or a Downstream Port of a Switch

Follow these steps to reach L0 at 8.0 GT/s:

1. Follow the steps in Section A.2.2.1 to reach Recovery.RcvrLock at 2.5 GT/s.

2. The PTC transmits TS1s with the non-PAD link and lane numbers set during the Configuration states. The speed_change bit is set to 1. All three speeds are advertised. After receiving 8 TS1s or 8 TS2s with the identical link and lane numbers to the ones being transmitted the PTC transitions to Recovery.RcvrCfg.

3. The PTC transmits TS2s with non-PAD link and lane numbers. The PTC shall let the DUT start at 8.0 GT/s using its own default Tx preset settings and not send any preset requests. After receiving 8 TS2s with speed_change bit set to 1 and 8.0 GT/s advertised, and after at least 32 TS2s with speed_change bit set to 1 have been transmitted after receiving the first TS2 and without interruption by an EIEOS, both the PTC and the DUT transition to Recovery.Speed. If the 32 TS2s are interrupted by an EIEOS then the counting shall be reset and the 32 TS2s have to be retransmitted.

4. In Recovery.Speed the PTC transmits 1 EIOS and goes to electrical idle. The PTC switches to 8.0 GT/s data rate 0.5 ms after transmitting the EIOS. The next state is Recovery.RcvrLock at 8.0 GT/s data rate. Now the new data rate is 8.0 GT/s and Recovery.Equalization is entered through Recovery.RcvrLock and link equalization is performed as follows. The PTC shall start a counter with the first symbols being transmitted at 8.0 GT/s, to keep a track of the number of blocks being transmitted. This counter shall be used for scheduling SKP OS being transmitted.

5. The PTC enters Phase 0. The PTC transmits TS1s with EC = 00b, Tx equalization set to the presets it received in the EQ TS2s, and reflects its current coefficient values. After receiving 2 TS1s with EC = 01b within 2 ms of entering Phase 0, the PTC transitions to Phase 1. If 2 valid TS1s as described above are not received within 2 ms, the test is considered to be failing and the link falls back to 2.5 GT/s.

6. In Phase 1, the PTC transmits TS1s with EC=01b, preset set to the one requested in the EQ TS2s it received from the DUT and reflects its current coefficient values. After receiving 2 TS1s with EC = 10b within 2 ms it transitions to Phase 2. In Phase 1 it notes the values of the FS and the LF being sent by the DUT. If 2 valid TS1s as described above are not received within 2 ms, the test is considered to be failing and the link falls back to 2.5 GT/s.

7. In Phase 2 the PTC shall request appropriate valid coefficients. The PTC shall obtain new block alignment every time it receives the new coefficients. The PTC should see the new coefficients being reflected by within 1.5 μs in the bits 5:0 in symbol 7, 8, and 9 (byte 8, 9, and 10, respectively). After finalizing on the coefficients the PTC shall transition to Phase 3. The PTC may keep requesting different coefficients and choose the optimal one.

8. In Phase 3 the DUT might try to adjust the PTC's coefficients. After receiving 2 consecutive TS1s with EC = 11b, and coefficients that are different than the ones currently used by the PTC, the PTC shall switch to the new coefficients within 500 ns after receiving the new request in 2 consecutive TS1 and reflect them in bits 5:0 in symbol 7, 8, and 9 (byte 8, 9, and 10, respectively). If the coefficients are not valid, they shall be still reflected in the symbol 7, 8, and 9 but the PTC shall set the Reject Coefficient bit in bit 7 of symbol 9. The step 8 can be repeated multiple times by the DUT. During Phase 3 the PTC shall always transmit TS1 with EC = 11b. After receiving 2 consecutive TS1s with EC = 00b the PTC shall reenter Recovery.RcvrLock.

9. Now the PTC transmits TS1s at 8.0 GT/s data rate. The PTC sets speed_change bit to 0. After receiving 8 TS1 or 8 TS2 with identical link and lane numbers the PTC transitions to Recovery.RcvrCfg.

10. The PTC transmits TS2s. After receiving 8 TS2s with speed_change bit set to 0 and at least 16 consecutive TS2s have been transmitted after receiving the first TS2, the PTC transitions to Recovery.Idle.

11. The PTC shall now transmit a SDS Ordered Set and keep transmitting logical idle symbols 0x00. After receiving 8 symbol times of idle data and after at least 16 idle symbols have been transmitted after receiving the first idle symbol the PTC shall transition to L0.

A.2.4.2 DUT is an Add-in Card or an Upstream Port of a Switch

Follow these steps to reach L0 at 8.0 GT/s:

1. Follow the steps in Section A.2.2.2 to reach Recovery.RcvrLock at 2.5 GT/s.

2. The PTC transmits TS1s with link and lane numbers set during the Configuration states. The speed_change bit is set to 1. All three speeds are advertised. After receiving 8 TS1s or 8 TS2s with identical link and lane number to the one being transmitted the PTC transitions to Recovery.RcvrCfg.

3. The PTC transmits TS2s with non-PAD link and lane numbers. PTC can choose to transmit Presets using the EQ TS2 (bits 6:3 of symbol 6) and set the bit 7 of symbol 6 to 1. If not, the DUT will start the 8.0 GT/s using its own default settings After receiving 8 TS2s with speed_change bit set to 1 and 8.0 GT/s advertised, and after at least 32 TS2s with speed_change bit set to 1 have been transmitted after receiving the first TS2 and without interruption by an EIEOS, the PTC transitions to Recovery.Speed. If the transmission of 32 TS2s is interrupted by an EIEOS, then the counting shall be reset and the 32 TS2s have to be retransmitted.

4. In Recovery.Speed the PTC transmits 1 EIOS and goes to electrical idle. The PTC switches to 8.0 GT/s within 0.5 ms. The next state is Recovery.RcvrLock at 8.0 GT/s data rate. The PTC shall start a counter with the first symbols being transmitted at 8.0 GT/s, to keep a track of the number of blocks being transmitted. This counter shall be used for scheduling SKP OS being transmitted.

5. Now the new data rate is 8.0 GT/s and Recovery.Equalization is entered through Recovery.RcvrLock and link equalization is performed.

6. The PTC enters Phase 1 of the Recovery.Equalization state whereas DUT enters Phase 0. The PTC transmits TS1s with EC = 01b, Tx equalization set presets it feels appropriate to the trace length, and reflects its current coefficient values. After receiving 2 TS1s with EC = 01b within 2 ms the PTC should transition to Phase 2. If 2 valid TS1s as described above are not received within 2 ms, the test is considered to be failing and the link falls back to 2.5 GT/s.

7. In Phase 2 the DUT might try to adjust the PTC's coefficients. After receiving 2 TS1 with EC = 10b, and coefficients that are different than the ones currently used by the PTC, the PTC shall switch to the new coefficients within 500 ns and reflect them in bits 5:0 in symbol 7, 8, and 9 (byte 8, 9, and 10, respectively). If the coefficients are not valid, they shall be still reflected in the symbol 7, 8, and 9 but the PTC shall set the Reject Coefficient bit in bit 7 of symbol 9. The step 7 can be repeated multiple times by the DUT. During Phase 2 the PTC shall always transmit TS1 with EC = 10b. After receiving 2 TS1s with EC = 11b the PTC transitions to Phase 3.

8. In Phase 3, the PTC transmits TS1s with EC=11b, preset set to the one requested in the EQ TS2 it received from the DUT and reflect its current coefficient values. In Phase 3 the PTC shall request appropriate valid coefficients. The PTC shall obtain new block alignment every time it receives the new coefficients. The PTC should see the new coefficients being reflected by within 1.5 μs in the bits 5:0 in symbol 7, 8, and 9 (byte 8, 9, and 10, respectively). The PTC may keep requesting different coefficients and choose the optimal one. After finalizing on the coefficients the PTC shall transitions to Recovery.RcvrLock.

9. Now the PTC transmits TS1s at 8.0 GT/s data rate with EC = 00b. The PTC sets speed_change bit to 0. After receiving 8 TS1s or 8 TS2s with identical link and lane numbers the PTC transitions to Recovery.RcvrCfg.

10. The PTC transmits TS2s. After receiving 8 TS2s with speed_change bit set to 0 and at least 16 TS2s have been transmitted after receiving the first TS2, without interruption from EIEOS, the PTC transitions to Recovery.Idle.

11. The PTC shall now transmit a SDS Ordered Set and keep transmitting logical idle symbols 0x00. After receiving 8 symbol times of idle data and after at least 16 idle symbols have been transmitted after receiving the first idle symbol the PTC shall transition to L0.

B

Appendix B. Acknowledgements

- **AMD**
  - Gord Caruk
  - Dean Gonzales
  - Betty Luk
  - Steve Manning
  - Anthony Tam
  - Wayne Yun
- **Agilent**
  - Rick Eads
  - Gordon Getty
  - Thorsten Goetzelmann
  - Rob Vezina
- **IBM**
  - Will Atherton
  - Dustin Patterson
- **IDT**
  - Mike Chessin
- **Intel**
  - Dan Froelich
  - Manisha Nilange
  - Akshay Pethe
  - Marc Wells
- **LeCroy**
  - Linden Hsu
  - David Li
  - Joseph Schachner
- **LSI**
  - Richard Solomon
- **NVIDIA**
  - Raymond Barrios
  - Steve Glaser
  - Bill Simms
  - Mark Taylor
- **PLX**
  - Nagamanivel Balasubramaniyan
  - Swapnajit Mitra
  - Satish Venugopal
- **QuestTech**
  - Jerry Kinsley
  - Rich Minter
  - Dan Neal
- **Tektronix**
  - Sarah Boen
  - Jit Lim
  - Steve Reinhold
  - Kalev Sepp